Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

EARTH SCIENCES DIVISION

FMG, RENUM, LINEL, ELLFMG, ELLP, and DIMES: Chain of Programs for Calculating and Analyzing Fluid Flow through Two-Dimensional Fracture Networks—Theory and Design

D. Billaux, S. Bodea, and J. Long

February 1988

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

LBL-24914

FMG, RENUM, LINEL, ELLFMG, ELLP, and DIMES: Chain of Programs for Calculating and Analyzing Fluid Flow through Two-Dimensional Fracture Networks— Theory and Design

Daniel Billaux, Sorin Bodea, and Jane Long

Earth Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

February 1988

This work was supported by the Repository and Technology Program of the Office of Civilian Radioactive Waste Management of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Abstract

This report describes some of the programs developed at Lawrence Berkeley Laboratory for network modelling. By themselves, these programs form a complete chain for the study of the equivalent permeability of two-dimensional fracture networks.

FMG generates the fractures considered as line discontinuities, with any desired distribution of aperture, length, and orientation. The locations of these fractures on a plane can be either specified or generated randomly. The intersections of these fractures with each other, and with the boundaries of a specified flow region, are determined, and a finite element line network is output.

RENUM is a line network optimizer. Nodes very close to each other are merged, deadends are removed, and the nodes are then renumbered in order to minimize the bandwidth of the corresponding linear system of equations.

LINEL computes the steady state flux through a mesh of line elements previously processed by program RENUM. Equivalent directional permeabilities are output.

ELLFMG determines the three components of the permeability tensor which best fits the directional permeabilities output by LINEL. A measure of the goodness fit is also computed.

Two plotting programs, DIMES and ELLP, help visualize the outputs of these programs. DIMES plots the line network at various stages of the process. ELLP plots the equivalent permeability results.

Table of Contents

List of Figures

Acknowledgements

The authors thank Lea Cox and John Peterson for providing a thorough review of this report.

1.0 INTRODUCTION

Network models are useful tools for understanding the hydrology of fractured rock. Such studies of fracture hydrology can proceed by adopting a model for the network geometry, estimating the statistical distribution of the appropriate geometric parameters through field measurements, and generating realizations of statistically identical networks. Once the geometry of a particular realization is specified, flow through the network can be studied (Long et al., 1982). For example, one might use such a procedure to study the average equivalent permeability of a fracture network under various boundary conditions or as a function of scale of measurement (Long and Witherspoon, 1985). Such a study might be part of a larger study of hydrologic response to a perturbation such as the construction of an underground opening containing nuclear waste.

The advantage of the network model approach is that one is able to configure tests of system behavior in any manner desired. For instance, if we wish to know how the rock behaves under a regional gradient which is approximately linear, we can impose a linear gradient on the model. If we want to know the behavior as a function of scale, we simply change the size of the model. In the field one is normally constrained to the approximately radial flow imposed by a well test, and it may be difficult to know what volume is actually being tested and how that volume is likely to behave under different boundary conditions. On the other hand, numerical models must make assumptions about the fractures which may be false. For instance, network models usually assume that flow occurs in the fractures as between parallel plates, although in some cases, channeling may dominate the flow. Also, it is difficult to obtain appropriate data for the network geometry and especially the hydraulic conductivity of the individual fractures. Network modeling and in situ testing have complementary problems, and therefore the uses of these two techniques should be complementary. An appropriate phi-

losophy is to improve the network model at least until one can explain the in situ test results with the model.

Besides constituting a complete chain of programs for the study of the equivalent permeability of two-dimensional fracture networks, FMG, RENUM, LINEL and ELLFMG are also part of a broader set of programs capable of modelling flow and transport in both two-dimensional and three-dimensional line networks (Figure 1.1). FMG3D, DISCEL and DIMES' three-dimensional capabilities are documented in Gilmour et al. (1986a and b). CHANGE is documented in Billaux and Long (1988a and b). TRINET is documented in Karasaki (1987). The complete set of programs as described in Figure 1.1 is a unique tool for the modelling of flow and transport in complex two- or three-dimensional fractured rock geometries.

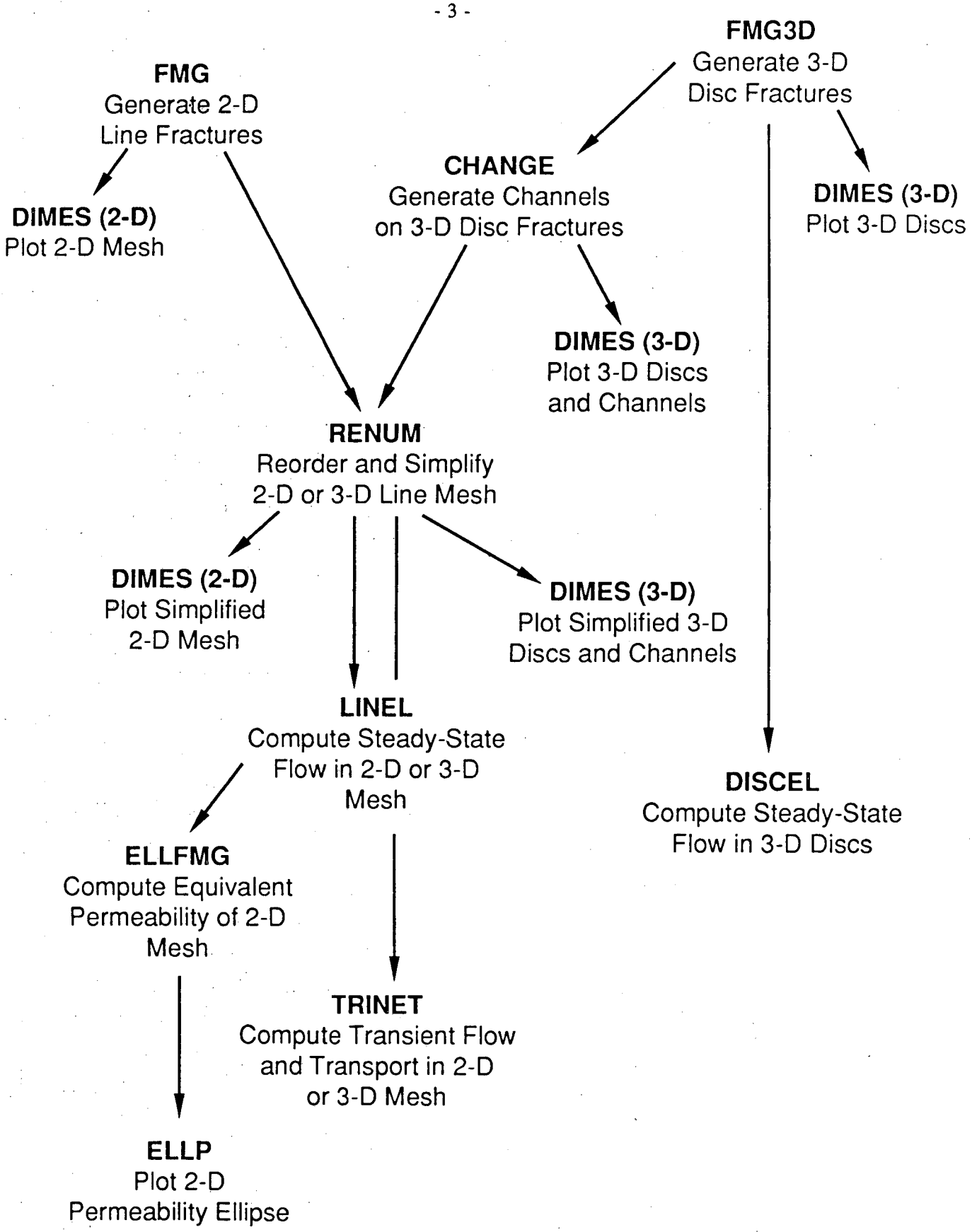


Figure 1.1. Programs for 2-D and 3-D modelling of flow through fractured rocks.

2.0 FRACTURE MESH GENERATION PROGRAM FMG

The purpose of program FMG is to create a fracture network input file for the optimization program, RENUM. To do this, FMG can either start with a specified fracture system or generate a random system based on stochastic variables provided by the user. By either process, the location, orientation, length, and aperture of all fractures in a generation region are determined. Next, FMG identifies only those fractures that are within a subregion called the flow region and are connected by at least one path to its boundaries. FMG determines the boundary nodes which are the intersections between fractures and flow region boundary lines and the intersections between fractures, which are called internal nodes. Non-conducting fractures may be eliminated from the fracture system. The geometric information needed to calculate flow is written into a file to be read by RENUM. RENUM will optimize the mesh. Flow through the fracture network of the flow region will be calculated by LINEL.

The first step is to generate a primary fracture system within a rectangular or circular region called the generation region. The fracture system is generated using one or more fracture sets. The number of sets, the number of fractures per set and the dimensions of the generation region [xgene, ygene] or [rgene] are specified by the user in the file FMG.INP. The fracture characteristics (coordinates of fracture center, orientation, length and aperture) may either be specified by the user or stochastically generated on a set by set basis.

Distribution functions that can be used to generate fracture orientation angles, lengths, and apertures currently include normal, lognormal, exponential, uniform, and normal with an option to correlate fracture aperture to fracture length. Fracture lengths are usually allowed to vary lognormally. The fracture apertures are usually generated assuming that apertures are lognormally distributed within a set. In general, field data will dictate which distribution function is appropriate for each fracture characteristic. The fractures extending beyond the boundary of

the generation region are truncated.

There are two options for stochastically generating the fracture mesh. In the first option ([nsgene]=0) the fracture network is generated set by set for the entire generation region such that the fracture centers are randomly located and uniformly distributed throughout it. In the second option ([nsgene]>0) the generation region is divided into a number of square or rectangle subregions, and the generation of the fracture mesh is made by subregions. In this case the fracture characteristics are either read in or generated for the first set and each subregion, second set and each subregion and so on, until the number of sets [nsets] is exhausted. The fracture characteristics are stored in the array [frac(nfrc,10)] to be used in later computations. The option of dividing the generation region into subregions and generating the fracture network by subregions was introduced to obtain a heterogeneous (i.e. "spatially variable") distribution of the mesh characteristics in the generation region.

Next, a rectangular or circular flow region is defined within the generation region. Input parameters that define the flow region include the two side lengths [xmesh, ymesh] and the orientation angle (θ), or the radius of the flow region [rmesh]. A circular hole can also be specified when a circular flow region is used. Its center [xhole, yhole] and radius [rhole] are specified by the user. The flow region is centered in the generation region. For a given primary fracture system, any number of flow regions of varying size and orientation or with different holes, may be defined and corresponding flow meshes generated.

The system of fractures lying within the flow region is determined by comparing the position of each fracture with the position of the four boundary lines or with the position of the flow and hole boundary circles. Fractures lying entirely inside the flow region are saved and fractures lying entirely outside the flow region are discarded. Fractures intersecting boundary lines are saved and the intersection(s) are stored as boundary nodes. A fracture may intersect more than one boundary line (depending on the geometry of the flow region, fracture location and length) and, therefore, contain more than one boundary node. Fracture lengths are truncated at boundary lines, and only the part falling inside the flow region and outside the hole (if

any) is used to solve for flow in the fracture network.

The system of fractures contributing to flow through the region is determined next. Intersections between fractures are found and stored as internal nodes and the segment between two consecutive nodes is defined as an element. Flow in a fracture can take place only through the elements of the fracture network so that in determining the fracture network for a steady state computation, any fracture containing less than two nodes cannot conduct flow and may, therefore, be discarded from the system of conducting fractures. The program starts with the fractures that intersect the flow region boundaries and determines the fractures they intersect. Then it considers the fractures it has just found and looks for new fractures intersecting them. This process goes on until there are no more fractures to be checked. As mentioned at each level the non-conducting fractures are discarded.

Finally, flow network information is prepared and written for input to the mesh optimization code, RENUM. Nodal arrays are assembled from stored boundary and internal nodes on conducting fractures in the flow region. All necessary run identification, fracture system, flow region, and nodal data are written to a file. This section describes the program FMG. The user's guide for the program as well as a listing of the code are provided in a separate report.

2.1 Generation of Fracture System

A primary fracture system, consisting of one or more sets of linear fractures randomly distributed within a rectangular or circular generation region, is created and fractures extending beyond the boundary are truncated.

2.1.1 Fracture System Characteristics

In the stochastic mode the primary fracture system is generated using parameters specified by the user. These include:

- the dimensions of the rectangular generation region, x_g and y_g([xgene],[ygene]), or r_g ([rgene])
- the number of fracture sets [nsets],

- either the number of fractures per set [nfrac] or the fracture density (i.e., number of fractures per unit area) per set [rlamb],
- distribution parameters for generating fracture orientation angles, lengths, and apertures.

The fracture centers are randomly generated within the region. The main program, FMG, calls subroutine FRAGEN to read input parameters and coordinate the generation of fracture characteristics. This section describes how FRAGEN works in the stochastic mode. Alternatively fracture centers, orientation angles, lengths, and apertures can be specified for any or all fracture sets.

Fracture Centers. The fracture centers are generated by either of two subroutines RECTXY and CIRCXY.

Subroutine RANDXY generates centers randomly distributed throughout a rectangular region. Coordinates of fracture centers (x_c, y_c) are computed by generating doublets of random numbers, uniformly distributed between zero and one, then scaling them by multiplying by the length of the generation region [xgene] ([ygene]), and subtracting 1/2 [xgene] (1/2 [ygene]). This makes the center of the generation region the origin of the rectangular coordinate system.

Subroutine CIRCXY generates centers randomly distributed throughout a circular region. The coordinates (x_c, y_c) are computed by generating doublets of random numbers (a, b), uniformly distributed between zero and one, subtracting 0.5 from them, and then keeping only the pairs of numbers such that

$$a^2 + b^2 \leq 0.25$$

a and b are then scaled by multiplying them by twice the radius of the generation region 2*[rgene].

Orientation. The orientation of each fracture line is determined by the angle which the fracture forms with the x-axis of the generation region. Fracture orientation angles are either input or generated by the program. If statistically generated, the mean orientation of each fracture set is required as well as other parameters, depending on the distribution chosen.

Fracture Lengths and Apertures. The length and aperture of each fracture are generated according to a normal, lognormal or exponential distribution. The generation procedure for the first two distributions requires the mean [ev] and standard deviation [sd] be specified for each fracture set. The exponential distribution requires only the mean. Apertures may also be correlated with fracture lengths (see the use of subroutine NORMD1).

Statistical Simulation

Random number generator. The statistical distribution subroutines and RANDXY use a random number generator called GGUBFS which is an International Mathematical and Statistical Library (IMSL) subroutine. GGUBFS returns random numbers uniformly distributed between zero and one and requires a double precision seed value [dseed]. GGUBFS returns a different random number each time it is called within a program. However the same sequence of random numbers is produced each time the program is run with the same initial seed. This mode of operation is optionally overridden by generating an arbitrary seed,

$$\text{dseed} = \text{SECNDS}(0.0) * 100.0. \qquad (2\text{-}2)$$

where SECNDS is a VAX-11 FORTRAN function subprogram which returns the system time of day in seconds less the value of its argument. An input flag [iranf] controls whether the seed is read or generated, and the initial seed is printed out. Since the seed defines the starting location for the random number generator, the user can reproduce a series of random numbers, i.e., reproduce a random fracture system by inputting the same initial seed in a later run.

Random generation of fracture centers. Depending upon the value of the input parameter [igene], subroutine RECTXY or CIRCXY calls GGUBFS once for each coordinate of the fracture center, to get two numbers a and b. The coordinates are then computed from a and b using either:

$$x_c = \text{FLOAT}[\text{INT}(x_g * 10^n * a)]/10^n - x_g/2$$
$$y_c = \text{FLOAT}[\text{INT}(y_g * 10^n * b)]/10^n - y_g/2$$

in the case of a rectangular generation region, or

$$x_c = \text{FLOAT}[\text{INT}(2 \cdot rg \cdot 10^n * (a - 0.5)]/10^n$$

$$y_c = \text{FLOAT}[\text{INT}(2 \cdot rg \cdot 10^n * (b - 0.5))]/10^n$$

in the case of a circular generation region. In the latter case, a pair (a, b) is used only if

$$a^2 + b^2 \leq 0.25$$

otherwise, a and b are discarded and GGUBFS is called again twice to get a new pair. In the equations above, n is the number of decimal places in the coordinate [itole], x_g is the length of the generation region in the x direction [xgene], y_g is the length of the generation region in the y direction [ygene], rg is the radius of the generation region [rgene], and d [dseed] is a double precision dummy variable initially equal to the input or generated seed then reset by GGUBFS. INT and FLOAT are intrinsic library functions which convert a real number to an integer by truncation and an integer to a real number, respectively. Truncating coordinates to n decimal places limits the minimum distance between fracture centers to the value 10^{-n}.

Random Generation of Fracture Characteristics

Normal distribution. In subroutine NORMAD, the sum S_N is calculated by calling GGUBFS twenty-five times and accumulating the sum,

$$S_N = \sum_{n=1}^{25} r_n \tag{2-4}$$

where r_n equals the value returned by a call to GGUBFS, r_n = GGUBFS(d). As shown by Hammersly and Hanscomb (1964), S_N is distributed normally with an expected value of 25/2 and a variance of 12/25; therefore,

$$S_N^* = \sqrt{\frac{25}{12}}\, S_N - \frac{25}{2} \tag{2-5}$$

is distributed normally with expected value 0 and variance 1. If μ and σ are the expected value and standard deviation supplied by the user ([ev] and [sd]) then

$$x = \sigma S_N^* + \mu \tag{2-6}$$

is distributed normally, N (μ , σ) with the specified parameters. (Note that in this equation, x does not refer to a point coordinate.)

Lognormal distribution. If S_N^* and x are defined as in the previous section, then S_N^* is distributed N(0,1), and x is distributed N(μ,σ), and y = exp(x) is distributed lognormally.

In terms of the parameters μ and σ of the normal distribution for x, the mean α and variance β^2 of the lognormal distribution are

$$\alpha = \exp(\mu)\exp(\sigma^2/2)$$

and

$$\beta^2 = \exp(\sigma^2+2\mu)[\exp(\sigma^2) - 1]. \tag{2-7}$$

Since the user will specify α and β, it is necessary to solve for μ and σ in terms of these variables:

$$\mu = 2 \ln \alpha - \frac{1}{2} \ln (\beta^2 + \alpha^2), \tag{2-8}$$

$$\sigma = \sqrt{\ln (\beta^2 + \alpha^2) - 2 \ln \alpha}.$$

Therefore, subroutine LOGNOD can calculate y from

$$y = \exp(\sigma S_N^* + \mu), \tag{2-9}$$

where μ and σ are defined above and α and β are specified by the user ([ev] and [sd]).

Exponential distribution. In subroutine EXPOND, μ is the expected value given by the user [ev] and

$$x = \mu \ln(1 - r) \tag{2-10}$$

is distributed exponentially, where r = GGUBFS(d).

Normal distribution correlating two variables. Subroutine NORMD1 generates random variables, x, distributed normally where the expected value, μ, is proportional to the logarithm of another parameter, x_1 or to the parameter itself. This correlation may be used to compute fracture aperture as a function of fracture length. S_N^* is defined as before (Equation 2-5) and the standard deviation, σ , is supplied by the user as [sd]. The user also supplies the y-intercept and the slope ([ycept] and [slope]) of a linear relationship between mean values of the variable x, and given values of the logarithm of x_1. The expected value of x, μ is computed,

$$\mu = \text{ycept} + \text{slope}*\log_{10} (x_1) \tag{2-11}$$

and x is computed as before, (Equation 2-6).

Uniform distribution. Subroutine UNIFOD generates random variables, a_i, distri-

buted uniformly over a given range, ainf to asup, such that:

$$a_i = (asup - ainf)*GGUBFS(dseed) + ainf \quad (2\text{-}12)$$

Additional distributions. The fracture generation code can easily be modified to include additional distribution functions that are found to be appropriate for any of the fracture characteristics.

2.1.2 Equations of Fracture Lines

The coefficients of the equation of the line in which a fracture lies are computed by subroutine EQLINE from the fracture's orientation and the coordinates of the fracture's center. The equation of a line is ax + by + c = 0, where a = -sin(orie); b = cos(orie) and c = - b * yc - a * xc are the coefficients of the line with a given orientation [orie] and passing through the fracture center of coordinates (xc,yc).

2.1.3 Truncation at Boundary

Fractures extending beyond the generation region are truncated at the boundary by the subroutine LIMIT. If a fracture is truncated the coordinates of the end points and the length of the fracture are recalculated. The fractures lying inside the generation region are used for statistical calculations and to determine the flow for a given study region.

2.1.4 Statistical Calculations

At the end of the generation stage simple statistical calculations are performed. This task is done by the subroutine PFS which calculates and prints the fracture statistics elements ([sd] and [ev]) for orientation, length and aperture in order to compare the generated fracture system with the input parameters provided by the user.

These statistics are computed two other times in the program, when the fracture network is altered.

2.2 Fracture System in the Flow Region

The flow region, which is the part of the fracture network to be analyzed, is included in the generation region and only fractures within this region are considered for input to the fluid flow model. For one random generation of fractures, several flow regions can be specified. They will be processed sequentially, and a finite-element mesh will be built for each of them. Fractures extending beyond the flow region are truncated at the boundary and those lying entirely outside the region are discarded. Intersections between fractures and flow region boundary lines are identified at this time. These intersection points, called boundary nodes, will be assigned fixed head or fixed flux in subroutine WRENUM.

2.2.1 Flow Region

The flow region is either defined by a rectangle centered at the origin of the generation region ([igene] = 0), or defined by a circular outer boundary and a circular innerhole ([igene] = 1). The dimensions, [xmesh], [ymesh], and the orientation angle θ [theta] of the rectangle, or the radius [rmesh] of the outer boundary and the radius [rhole] and position [xhole, yhole] of the hole are input parameters read from FMG.INP. They must be such that the resulting region lies completely within the generation region. The next step is to determine which fractures of the primary fracture system lie within the flow region.

2.2.2 Fractures in Flow Region - Rectangular Case

Subroutine RLIMIT truncates the fractures intersecting the rectangular flow region boundaries and discards those falling completely outside of it. If a certain fracture is truncated, the subroutine RLIMIT will recalculate the coordinates of the end points and the length of the truncated fracture. The information thus obtained about fractures falling within the flow region will be stored and used to determine the flow mesh.

We mentioned that the flow region is centered in the generation region. Then the equations of the boundary lines will be:

$$Ax + By \pm C = 0$$

$$Bx - Ay \pm D = 0$$

where C = ymesh/2 and D = xmesh/2.

Using the equation of the line that passes through each fracture or "fracture line", we then determine its intersection with the four lines that lie along the flow region boundaries. Using the coordinates of the first endpoint of the fracture, the two or four intersections are parameterized using the parameter t [t(4)] such that:

t = 0 at the first endpoint of the fracture

t = length of the fracture at the other endpoint

The number of intersections, either 2 or 4, is stored in variable [nt]. From Figure 2.1, one can see that the part of the fracture that falls inside the flow region has to lie between the two middle intersections. The array [t] is ordered. Then in order to determine if the fracture line passes through the flow region, the first two values in the ordered [t] array are considered. If they correspond to two adjacent sides, then the fracture line passes through the flow region. If they correspond to two opposite sides, then the fracture line does not intersect the flow region (Figure 2.2), except in the special case of the fracture being parallel to boundaries. In this case, the fracture is discarded only if its distance to the center is greater than the relevant flow region size [xmesh]/2 or [ymesh]/2 (Figure 2.3). Since the sides of the flow region are numbered consecutively, the sum of the numbers of two adjacent sides is odd, whereas the sum of the number of two opposite sides is even. The side numbers for each intersection are stored in the array [is(4)]. So we want to keep all fracture lines with four intersections (i.e. non-parallel to a side) and an odd sum is(1) + is(2), and all fracture lines with two intersections (i.e. parallel to a side). This is achieved by keeping only fracture lines for which the sum [is(1) + is(2) + nt/2] is odd.

At this point we have determined if a fracture line passes through the flow region. We also know which segment of the fracture line is in the flow region. It is the segment t1-t2, with t1 = [t(1)] and t2 = [t(2)] when [nt] = 2, and with t1 = [t(2)] and t2 = [t(3)] when [nt] = 4. We must now determine which part of the fracture overlaps this segment. If all the fracture

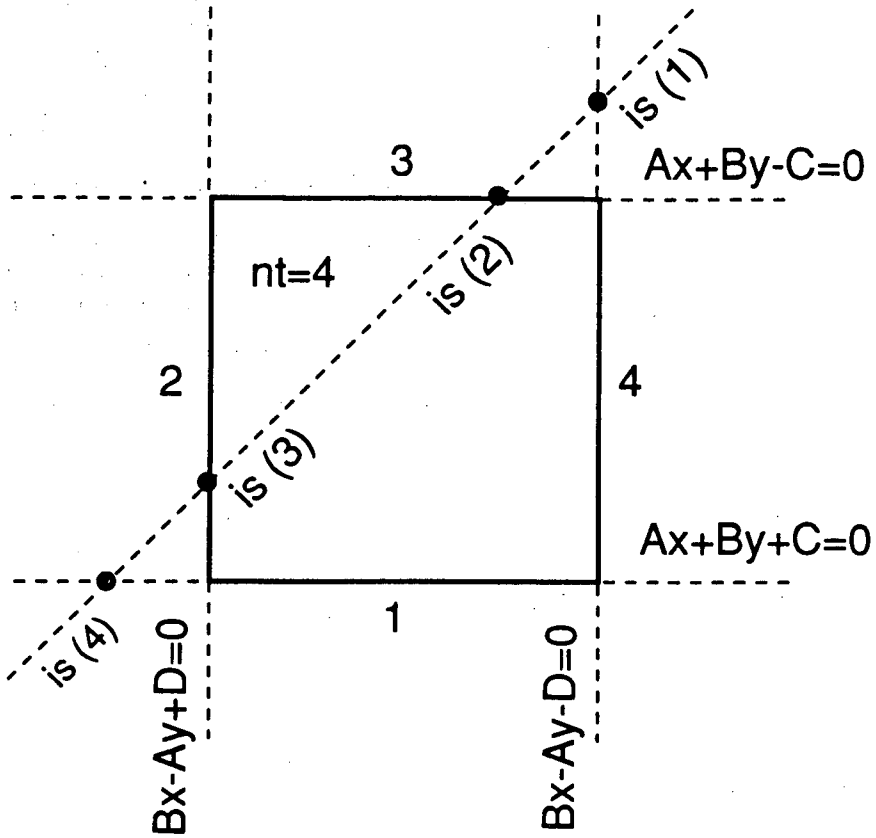


Figure 2.1. Part of a fracture line inside the flow region.

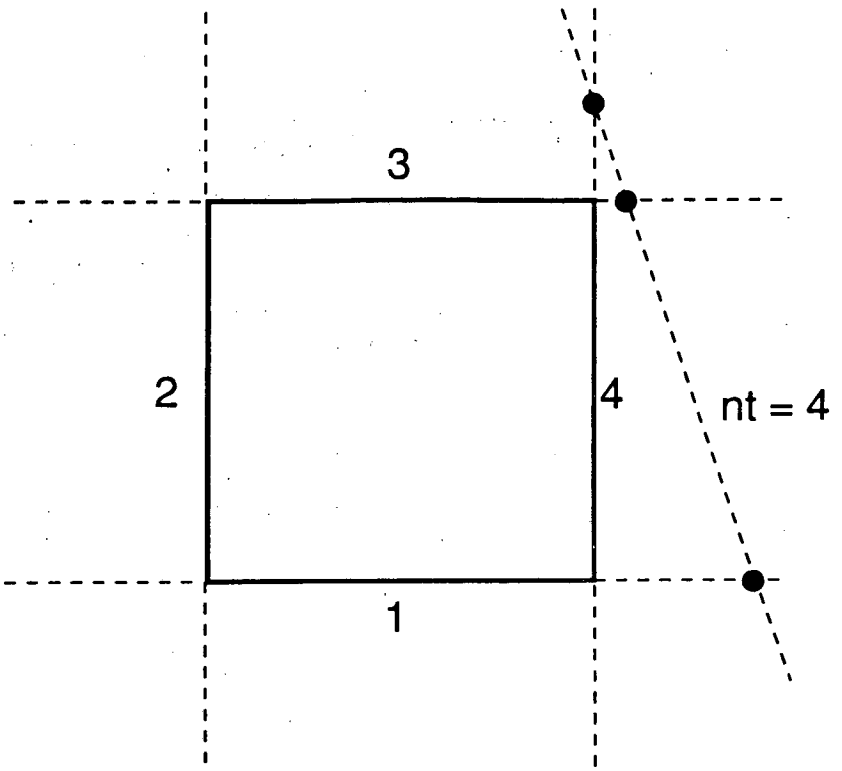


Figure 2.2. Checking side numbers to determine if a fracture line passes through the flow region.

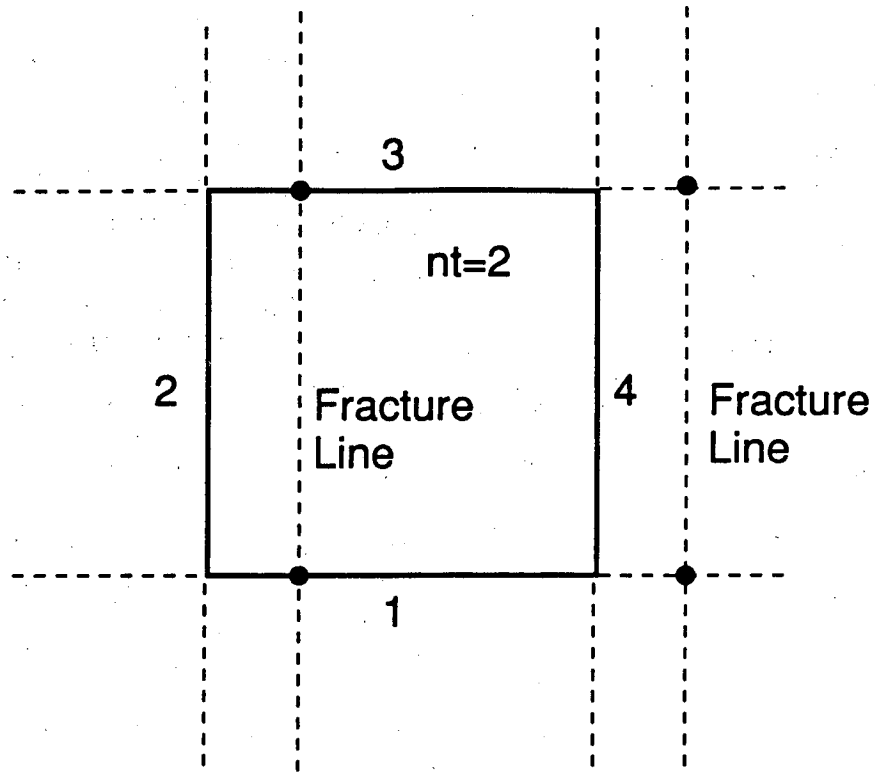


Figure 2.3. Discarding fracture lines which are parallel to the boundaries.

is inside the segment, we keep it. If only part of the fracture is inside the segment, we truncate this fracture. Finally, if the fracture and the segment do not overlap, we discard the fracture. If t1 > fracture length or t2 < 0 (Figure 2.4) the fracture is outside the flow region. Otherwise, the fracture is kept, and truncated if needed (Figure 2.5). The new fracture information is stored in array [frac].

2.2.3 Fractures In Flow Region - Circular Case

Subroutine CLIMIT truncates the fractures intersecting the circular flow region boundaries and discards those falling completely outside of it. If a certain fracture is truncated, the subroutine CLIMIT will recalculate the coordinates of its endpoints and the length of the truncated fracture. A fracture may also be split in two parts by the inner hole. In this case, two fractures with the proper endpoints and lengths are created from the generated one. The information thus obtained about fractures falling within the flow region is stored and used to determine the flow mesh. A do loop over the generated fractures is performed. After its proper endpoints and length have been recomputed to account for eventual truncation, the characteristics of a fracture are rewritten in the same array [frac]. However, the identification number of the fracture (i.e. its position in the array [frac]) may change after it has been examined, because whenever a fracture must be discarded, the counter for writing characteristics into [frac] is not incremented, so that the next fracture will be written over it in the array.

First the fracture is truncated at the perimeter of the outer flow region circle. Since the origin of the coordinates is its center, the equation of the flow region circle is:

$$x^2 + y^2 = r_m^2$$

where r_m [rmesh] is the radius of the circle. We parameterize the line supporting the fracture by the relative distance t from the first endpoint (x,y):

t = 0 at the first endpoint (x_1, y_1)

t = 1 at the second endpoint (x_2, y_2)

then, if alen is the length of the fracture, the distances t1 and t2 for the intersections of the line

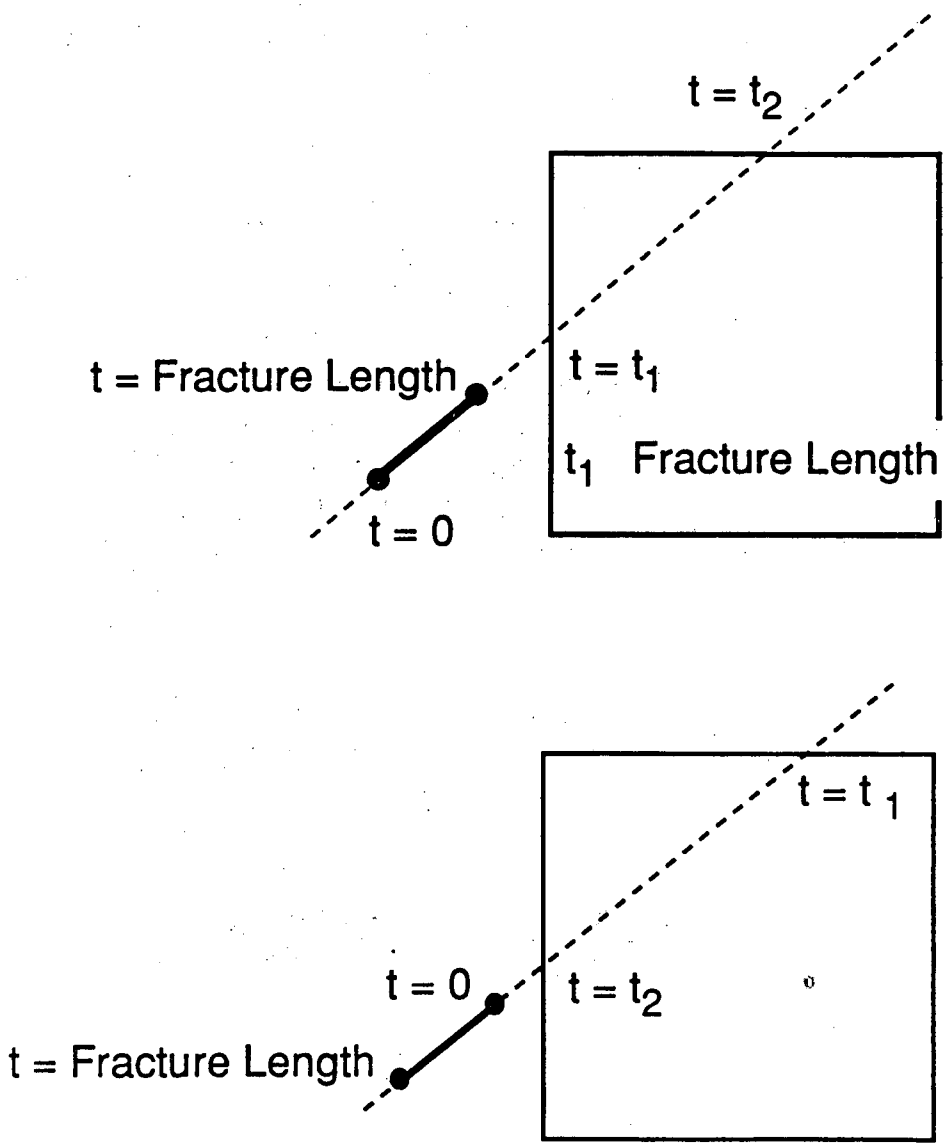


Figure 2.4. Using array [t] to discard fractures which fall outside the flow region.

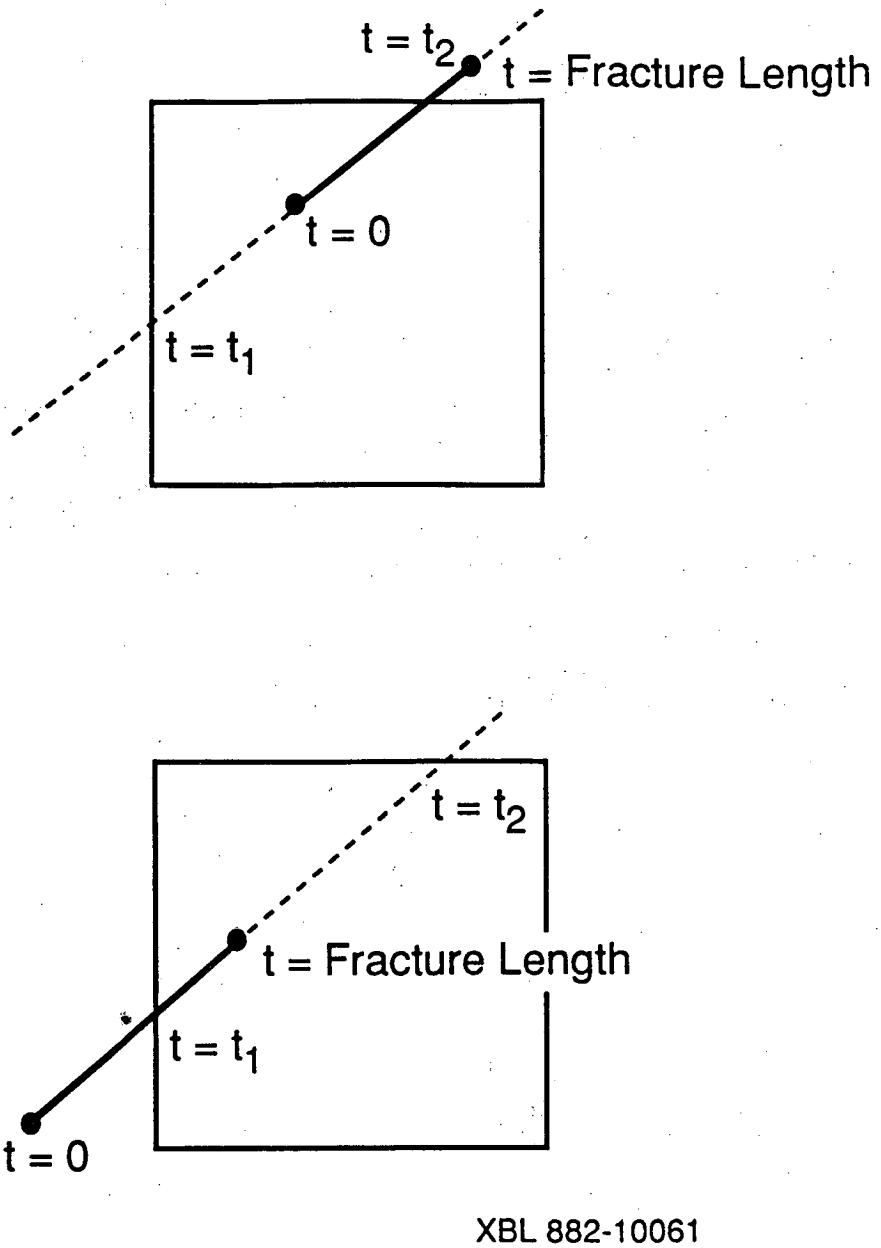


Figure 2.5. Truncation of fractures which fall both inside and outside the flow region.

with the circle are the solutions of:

$$alen^2 t^2 + 2[x_1(x_2-x_1)+y_1(y_2-y_1)]\ t+x_1^2+y_1^2-r_m^2 = 0 \qquad (2\text{-}13)$$

The fracture is at least partly inside the flow region if

- the discriminant of equation 2-13 is positive (i.e. the fracture line cuts the flow region)
- t1 is less than 1,
- t2 is more than 0.

Otherwise it is discarded.

The first endpoint is truncated if t1 is greater than zero. The second endpoint is truncated if t2 is smaller than one. The values of x_1, y_1, x_2, y_2 and alen are modified accordingly if necessary.

Next the fracture is truncated or split at the inner circle, or "hole", with radius rh[rhole], and center (x_h, y_h) [xhole, yhole]. The coordinates of the endpoints are translated

$$xx_1 = x_1 - xh$$
$$yy_1 = y_1 - yh$$
$$xx_2 = x_2 - xh$$
$$yy_2 = y_2 - yh$$

Then the equation for the intersection(s) is exactly similar to equation 2-13:

$$alen^2 t^2 + 2(xx_1(xx_2-xx_1) + yy_1(yy_2-yy_{1)}]\ t + xx_1^2+yy_1^2 - rh^2 = 0 \qquad (2\text{-}14)$$

This time we want to discard anything inside the circle.

So a fracture will be kept unchanged at this step if:

- the discriminant of equation 2-14 is negative (i.e. the fracture line does not cut the circle)
- t1 is more than 1
- t2 is less than 0

A fracture will be discarded if t1 is less than zero and t2 is more than one (fracture totally included in the hole). If either t1 is more than zero or t2 is less than one, then the second endpoint or the first endpoint respectively is truncated, so the endpoint coordinates and length are updated. If both t1 and t2 are between zero and one, then the fracture must be split in two fractures. This is done by simply assigning to the current fracture number the first part of the fracture, and then by considering the second part of the same fracture, instead of the next fracture, when incrementing the fracture number. This requires that the number of the next generated fracture to be processed be higher than the number of the current truncated fracture plus one. This may be true because some previous fractures have been discarded. If it is not true, subroutine MOVE is called, and translates all the characteristics of the fractures not yet studied down in the array [frac]. The amount [idif] by which the characteristics are translated down is determined by [idif] = minimum ([nfrac]/10 + 1, [maxfrc] - [nfrac]), where [nfrac] is the number of fractures, and [maxfrc] is the size of the array [frac].

2.2.4 Statistical Calculations

After eliminating all fractures outside the flow region, and truncating all those which intersect the flow region boundaries, the same statistics as described in Section 2.1.4 are computed and printed by subroutine PFS. Comparison of these values with those computed for the generation region indicates whether the flow region is statistically representative.

2.2.5 Connections between Sides

When studying the percolation properties of networks, one important parameter is the percolation threshold. This is the "minimum" set of statistical properties for which a connected network of infinite size (infinite cluster) exists. Below the threshold, only finite-size clusters of fractures exist. In order to compute this threshold, we generate large networks and simply check if there is a connection between sides 1 an 3 (rectangle), or between the center hole and the outer boundary (circle). The ability to check connections this way without outputting a finite element network has been added to the program. This is performed by subroutine CON-

NEC if the flag [icont] is set to 5 in the input deck.

2.3 Fracture System to be Used in Flow Model

In order to calculate flow through the system, all fracture intersections must be located. Intersections between fractures and boundary lines have already been determined. Therefore, the next step is to locate all intersections between fractures (internal nodes). This is done sequentially, starting with fractures intersecting the boundaries, then fractures intersecting these ones, and so on, until no new intersection is found. In this way, all the fractures from which there exists no path to any boundary, i.e. isolated clusters, are automatically discarded. In the circular case, the search for intersections is initialized with only the fractures intersecting the hole. In this way, only fractures linked to the hole will be kept. They are the only ones having an effect when modelling a well test. For steady state problems, flow in a fracture can only occur between intersections. Therefore, only those fractures containing two or more nodes may conduct flow. To simplify the flow problem, all fractures containing less than two nodes are identified as nonconducting and eliminated from the catalogue. The elimination of dead-ends and isolated clusters can both be overridden to produce meshes for transient or fracture-matrix flow.

2.3.1 Calculation of Fracture Intersections

The fracture mesh is built by subroutine CONNEC from the rectangular boundaries to the inside of the flow region, or from the inner hole to the outside, level by level. Intersections of all the fractures intersecting the boundaries (fractures in level 1) with all the fractures, either in level 1 or not, are searched. All the fractures not in level one but which intersect a fracture in level one are put in level two, and are then screened for intersections, and so on. Fracture numbers are rearranged level by level during the process, by permutating elements of the [iold] reference array pointing from rearranged numbers to original numbers. When a fracture [ifrac] is studied, only fractures with numbers bigger than [ifrac] are screened for intersections. If the fractures intersect, the coordinates of the point of intersection are determined.

The following intersection information is stored: the identification numbers of intersecting fractures [ifrac(2,mnod)]; the distance from the first end point of the respective fracture [tint(2,mnod)]; the node numbers for each fracture [knode(mkey)]; the key to array [knode(mkey)] for each fracture [kut(mfrc)].

2.3.2 Elimination of Nonconducting Fractures

Depending on the value of the flag [ikeep] input by the user, non-conducting fractures are discarded or kept in. If [ikeep] = 0, simple dead-ends are discarded as well as isolated clusters. If [ikeep] = 1, dead-ends are kept but isolated clusters are still discarded. If [ikeep] = 2, then both dead-ends and isolated clusters are kept.

Discarding simple dead-ends [ikeep] = 0

Fractures with only one intersection are dead-ends for steady state flow. So at each level, once all the intersections of a given fracture have been found, if there is only one intersection, the fracture may be eliminated. Note that by removing the fracture, we are removing one intersection from a fracture in the previous level. This other fracture may then be left with only one intersection, and so on. The program goes back through the mesh deleting fractures until it finds a fracture left with two or more intersections. Note that this fails to get rid of dead-ends containing loops. These complex dead-ends will be discarded by RENUM.

Keeping isolated clusters [ikeep] = 2

Isolated clusters are discarded "by default" since the search for intersections is started at the boundaries and follows the connections. If [ikeep] = 2, when the search initiated at the boundaries is finished, the number of fractures already screened is tested against the total number of fractures in the flow region. If they are not equal, a fracture not yet included is just put arbitrarily in the next level, and the search is resumed starting from that fracture. The process is then repeated until all the fractures in the flow region have been searched.

2.3.3 Boundary Conditions for Rectangular Networks

The rectangular fracture networks produced by FMG are primarily used to compute directional conductivities. Such conductivities in several directions are then used to compute an equivalent continuous medium conductivity tensor, and to check if the continuum hypothesis holds by deriving a normalized mean square error. This is performed by program ELLFMG (Chapter 5). In a heterogeneous medium such as fractured rock, conductivity must be measured in the direction of the gradient. The average gradient can be constant in magnitude and direction throughout a heterogeneous region in steady flow if the region behaves like a homogeneous porous medium. The direction of flow, however, is controlled by the direction of the fractures. Since the direction of the gradient can be controlled, measuring permeability in the direction of the gradient is much easier than measuring in the direction of flow.

The boundary conditions necessary to produce a constant gradient in a rectangular anisotropic flow region are illustrated in Figure 2-6. They consist of two constant-head boundaries (ϕ_2 and ϕ_4) and two boundaries with the same linear variation in head from $\phi_1 = 1.0$ to $\phi_4 = 0$. Conductivity is measured in the direction perpendicular to sides 2 and 4.

The linearly varying boundary conditions in sides 1 and 3 are necessary because, in general, the medium in the flow region is anisotropic. Without these boundaries, the lines of constant head would be distorted near sides 2 and 4 as shown in Figure 2-7. When the isopotentials are distorted, only part of the flow region can experience a constant gradient. In an arbitrary heterogeneous system of unknown anisotropy, it is impossible to determine which part of the system is experiencing a constant gradient and which part is not. Therefore when no flow boundaries are used, it is not always possible to measure only that part of the flux which is due to a known constant gradient.

The boundary conditions used in Figure 2-6 insure that the whole fracture system is equally stressed by the hydraulic gradient. Under these boundary conditions, the existence of a constant gradient in the flow region depends only on how well the fracture system is interconnected. If the system is well connected, it will behave like a porous medium and have a

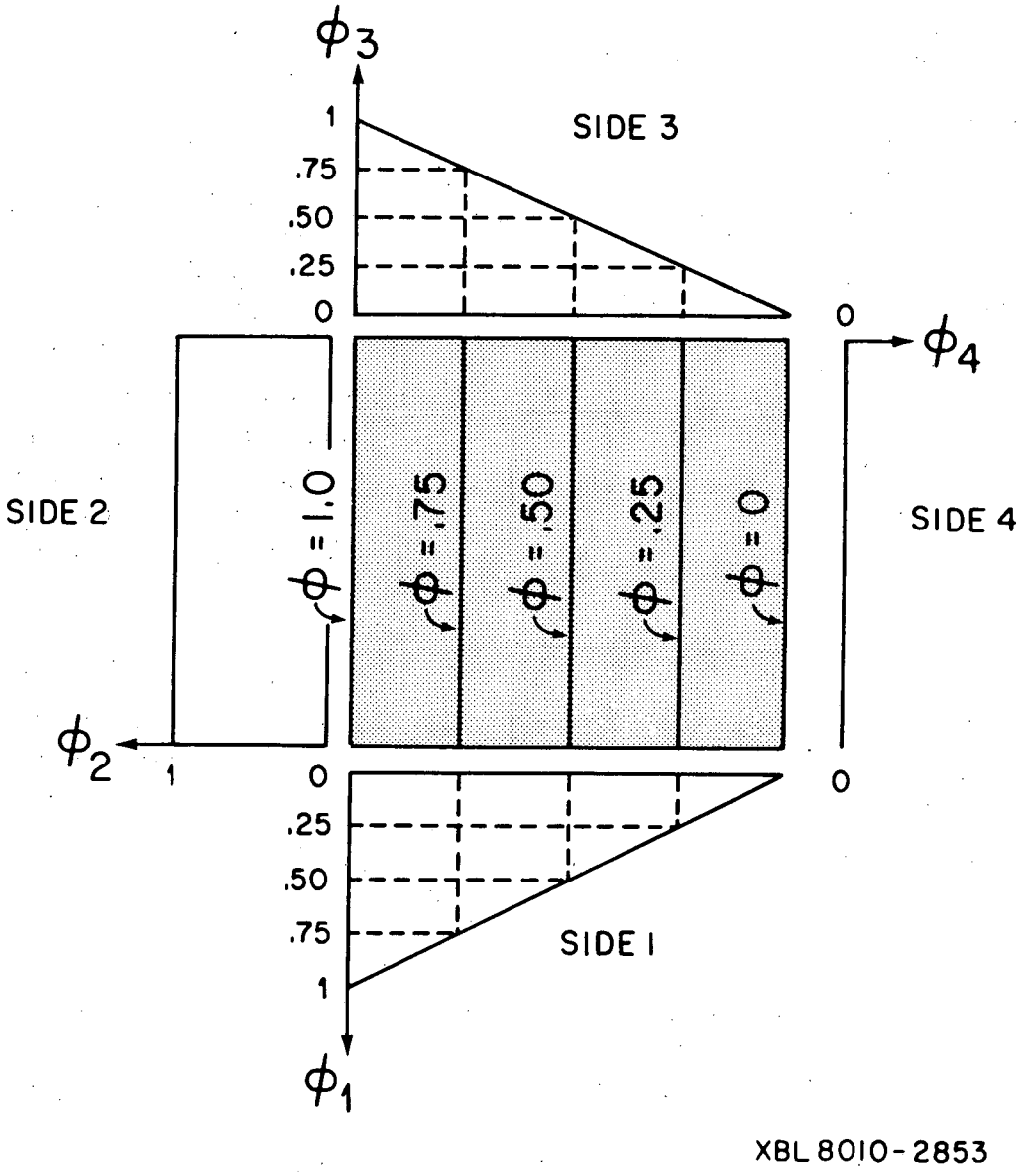


Figure 2.6. Boundary conditions applied to fracture models for permeability measurement.

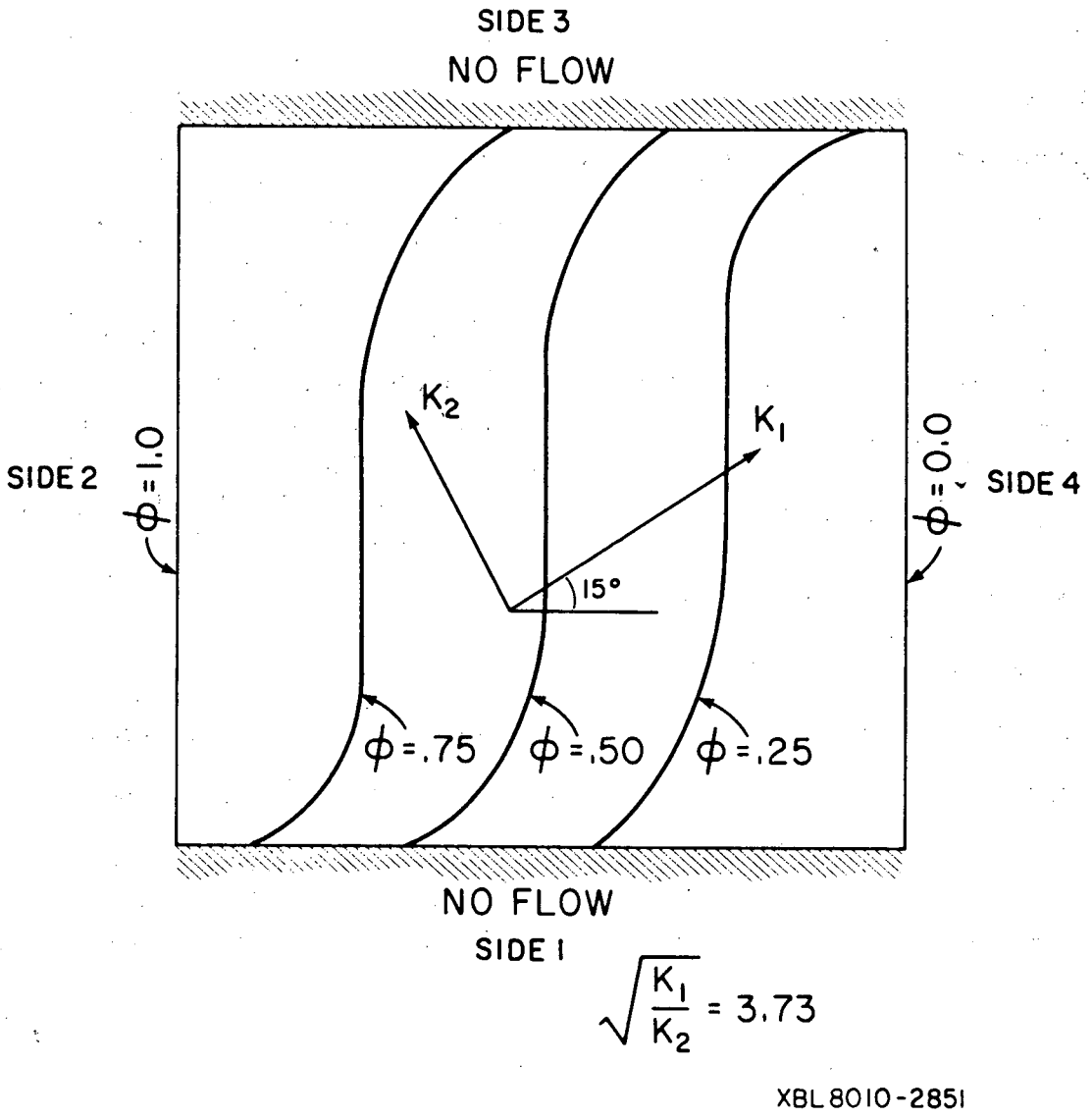


Figure 2.7. Distortion of isopotentials in an anisotropic medium with "no flow" boundaries.

constant gradient.

In order to accommodate directional conductivity studies, FMG can handle both constant and linearly varying imposed head boundary conditions. An option is also provided to produce constant flux boundary conditions. Note that if such conditions are used, the processing will end with program LINEL (Chapter 4), and program ELLFMG will not be able to compute an equivalent porous medium conductivity tensor.

2.3.4 Boundary Conditions for Circular Networks

The circular networks produced by FMG are primarily used to simulate well-tests. A constant flux or constant head is imposed on the inner hole representing the well, and constant head conditions are imposed at the perimeter of the flow region. This reproduces the conditions during a well test.

2.3.5. Finite Element Mesh

Subroutine WRENUM reads the boundary conditions specifications for a given flow region and computes the imposed head at each boundary intersection if a linearly varying imposed head is specified. The following specifications for each node are then output:

- the identification number of the node
- a code identifying if the node is internal, or a boundary with imposed head, or a boundary with imposed flux (0, 1, -1)
- the side number for a boundary node (1, 2, 3, 4 if rectangular; 1,3 if circular)
- the coordinates of the node
- the value of the imposed head(s) or flux(es) at the node if needed

Finally, the element catalogue is printed, including:

- the element identification number
- the numbers of the two nodes at the endpoints of the element

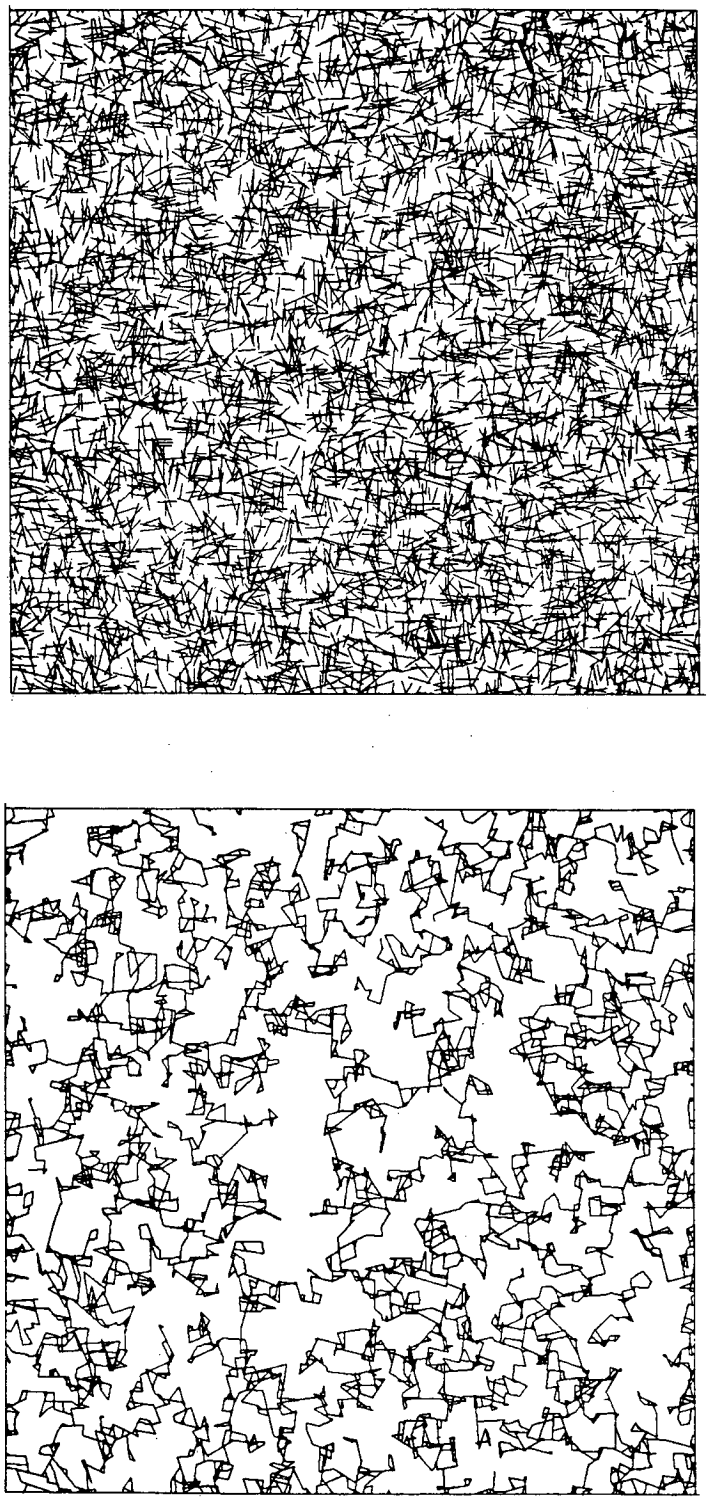
- the transmissivity of the element
- the length of the element
- the identification number for the fracture on which the element lies (this is printed only for reference and not used by the next program).

3.0 PROGRAM RENUM

The program RENUM was designed to better use the memory space and reduce the computing time necessary in the simulation process of the hydraulic behavior of fractured rock. It simplifies fracture networks for a more efficient computation. The code merges the nodes which are too close to each other, removes all dead-end clusters which do not conduct fluid and renumbers the mesh in an efficient way. RENUM is a part of a chain of programs developed at LBL, and can treat the output of FMG or CHANGE (Billaux and Long, 1988).

In two dimensions (Figure 3.1), fractures are assumed to be line segments (Long, 1983), and are generated by FMG as explained in Chapter 2. The chain of programs outlined in Figure 1.1 also includes three-dimensional codes. FMG3D generates random disc (Gilmour et al., 1986) networks. The discs can be used by program CHANGE (Billaux and Long, 1988) as the supports on which random channels are generated (Figure 3.2). The channels generated on the discs constitute a three-dimensional network of line elements. The problem of computing flow and transport in such networks is numerically exactly the same as computing the flow in two-dimensional line-segment networks (however the number of nodes and elements involved may vary considerably). RENUM, initially built to process two-dimensional networks, can therefore also handle three-dimensional channel networks generated by CHANGE.

In steady state, the flow problem is fully defined after the conditions on the boundaries have been given. We define the nodes as the intersections between line elements or between a line element and a boundary. Solving for the flow in each line element and the potential head at each node amounts to solving a system of linear equations, the number of unknowns being the number of nodes.



XBL 881-247

Figure 3.1. Two-dimensional fracture mesh. (a) fractures generated pseudo-randomly, (b) same mesh with simple dead-ends removed.

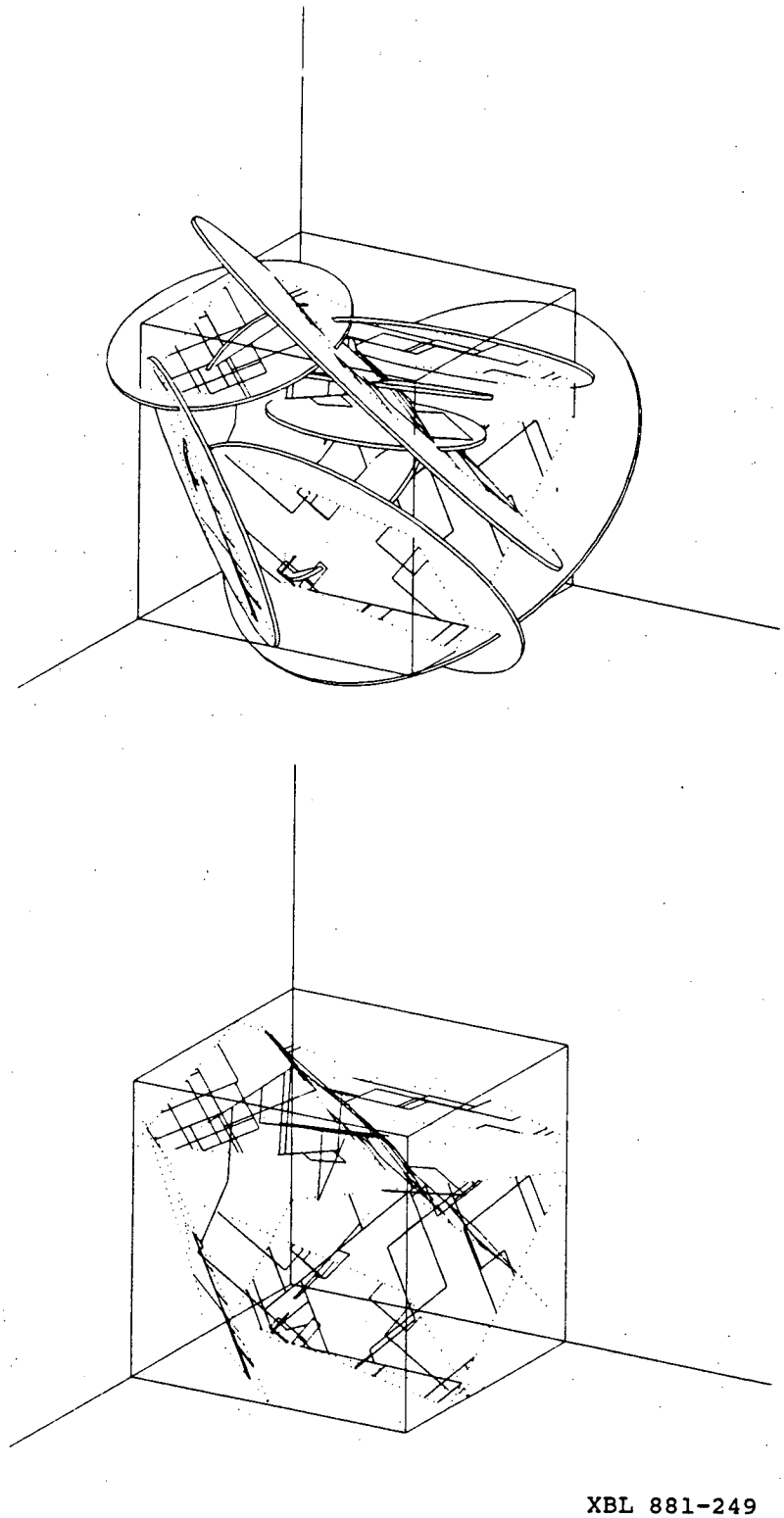


Figure 3.2. Three-dimensional channelized fracture mesh. (a) random discs and channels, (b) channels only.

3.1 Merging Endpoints of Short Elements

Due to the fact that large differences between the lengths of the elements can lead to an ill posed problem (the matrix to be solved is not positive definite within the precision of the computer), very short elements are discarded and their ends are merged. This is done in sub-routine MERGE.

In the discarding process we use the three arrays inode(2,i), inew(i) and ibs(2,i). The program loops over the elements and compares their endpoints. If the coordinates of these two nodes are equal then one of them is discarded as follows:

1. If the two nodes i and j that form the element are both boundary nodes, node i is discarded by setting inew(i) = j. In this way, any later reference to node i will be pointed instead to node j.
2. If one node i is a boundary node and the other j is not, node j is discarded by setting inew(j) = i.
3. If both nodes i and j are interior, then node j is discarded (inew(j) = i).

In this way, no connection to the boundaries is cut.

When an inner boundary condition with imposed flux is specified, all the nodes on this boundary are merged. In this way, the inner hole is modelled as an infinite permeability medium.

3.2 Discarding Dead-ends

Some definitions from graph theory are useful in describing the algorithm used to discard dead-ends (Billaux and Fuller, 1988). A connected graph is one in which any two nodes are connected to each other by at least one path. In our case, this path can be through fractures or boundary lines or both. By extension, a biconnected graph is one in which any two nodes are connected to each other by at least two totally distinct paths. In a given graph, a biconnected component is a maximum biconnected sub-graph, i.e., a part of the graph as big as possible while still being biconnected. It is connected to the rest of the graph by only one node, which

is then called an articulation point.

We want to keep any node which is connected to the boundaries by at least two distinct paths as this node will be on a connected flow path. Selecting the "active mesh" then amounts to selecting all biconnected components containing at least two boundary points. The algorithm used in subroutine PATHS identifies and lumps together all the biconnected components including two boundary nodes, and does not attempt to distinguish between the other ones, thus saving some time. To achieve this, a breadth-first tree search is performed starting from all boundary nodes as initial points, as explained below. Figure 3.3 shows a simple mesh used to illustrate the description of the algorithm.

The breadth-first search is a simple way of examining once every node connected to the initial points. It is performed iteratively by scanning the nodes in successive levels defined as follows. Level one contains the initial nodes. Level two contains all nodes connected to level one by a line element. Level n + 1 contains all the nodes connected to a node in level n by a line element and which are not already in level n or n - 1. Note that a node connected to a node in level n by a line element cannot be in a level lower than n-1. Otherwise the node in level n would have to be in a lower level.

The process is initialized by putting in level one all the initial nodes. Then a loop over the levels is started. For any level, each node in the level is studied to find all the nodes connected to it. If a new node is encountered, it is put in the next level. At that time, we call the node we are examining the "study node." Once all the nodes of a given level have been screened, the loop over the levels is incremented and screening of the nodes in the next level begins. This goes on until at the end of the screening of a level the next level is empty. During the search, the nodes are numbered sequentially as they are encountered. In this process we do not encounter any nodes which are isolated from the boundary, therefore, these nodes are automatically not included in the catalogue.

The boundary nodes are the initial points or "sources" for the first search. A sufficient condition for any node to be active is to be reached by two branches of the search originated

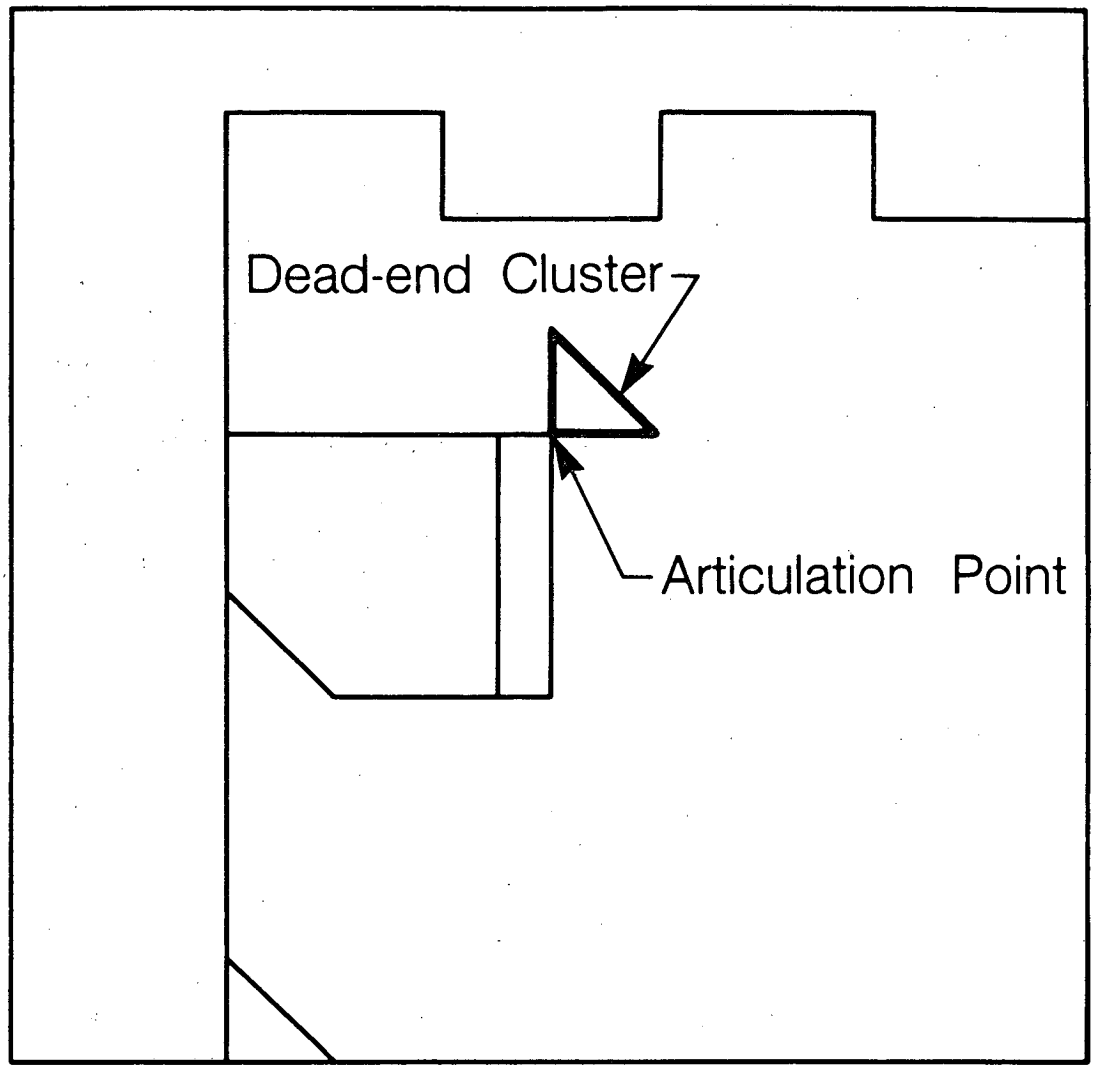


Figure 3.3. Simple mesh with dead-end highlighted.

from different sources. For any node we encounter, we remember on which branch it is by recording the source from which this branch originated. Therefore, the following rules are followed when studying a given node during the search and looking for all nodes connected to it by a line element:

- If a new node is encountered, we record the number of its source. This number is in fact the source number of the current study node from which we have just found the new node. One exception is the case when the study node is a source node. In this case the source of the new node will be the study node itself. The new node is also added to the list of nodes in next level.
- If the node we encounter has been found earlier in the search, it already has a source number. We compare this source number with the number ns we would like to assign to it if the node was new. If they are identical, both the study node and the node we have encountered are part of the same branch of the search tree and the search goes on. On the contrary, if the two sources are different then we have found two active nodes.

An active node is flagged by setting its source to zero. When this node is studied later in the search, it will be considered as a source itself, so that its number will be the source number of any new node we will find connected to it. To enable an active node to act as a source, we need to find that it is active before it becomes the study node. Since we study the nodes in the order in which we renumber them, this will be the case if its number is higher than the number of the current study node. Once this is done, the search resumes.

Once the breadth-first search is terminated, we have completely renumbered the part of the mesh connected to the boundaries (Figure 3.4). The numbering is such at this stage that when considering any two nodes, the node with the lower number is either closer or as close to the boundaries as the node with the higher number. By closer, we mean that we need to pass through less line elements to go from the boundaries to the node. We can now derive a second sufficient condition for a node to be active. Consider a dead-end cluster (Figure 3.5).

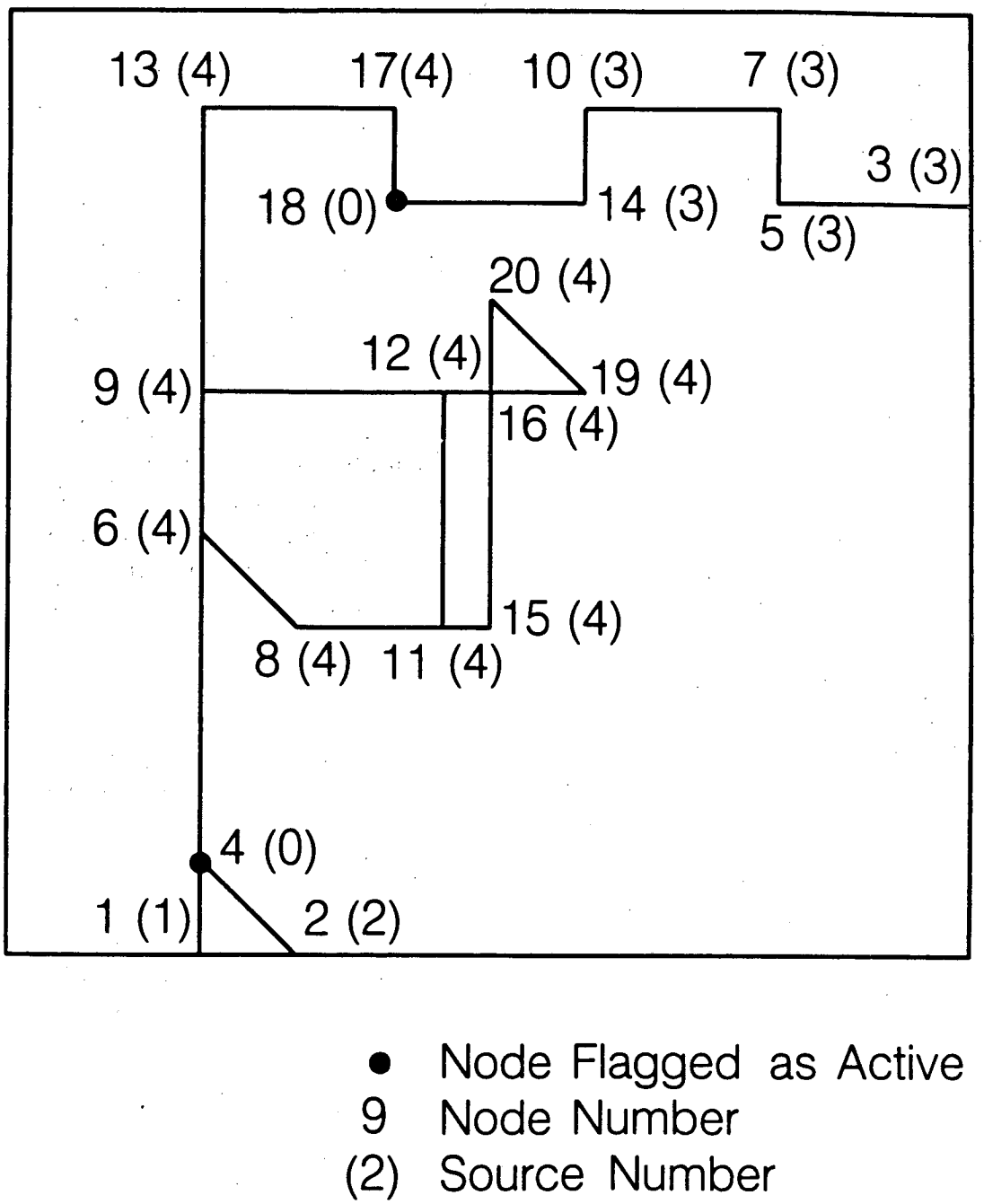


Figure 3.4. Mesh from Figure 3.3 at the end of the first search.

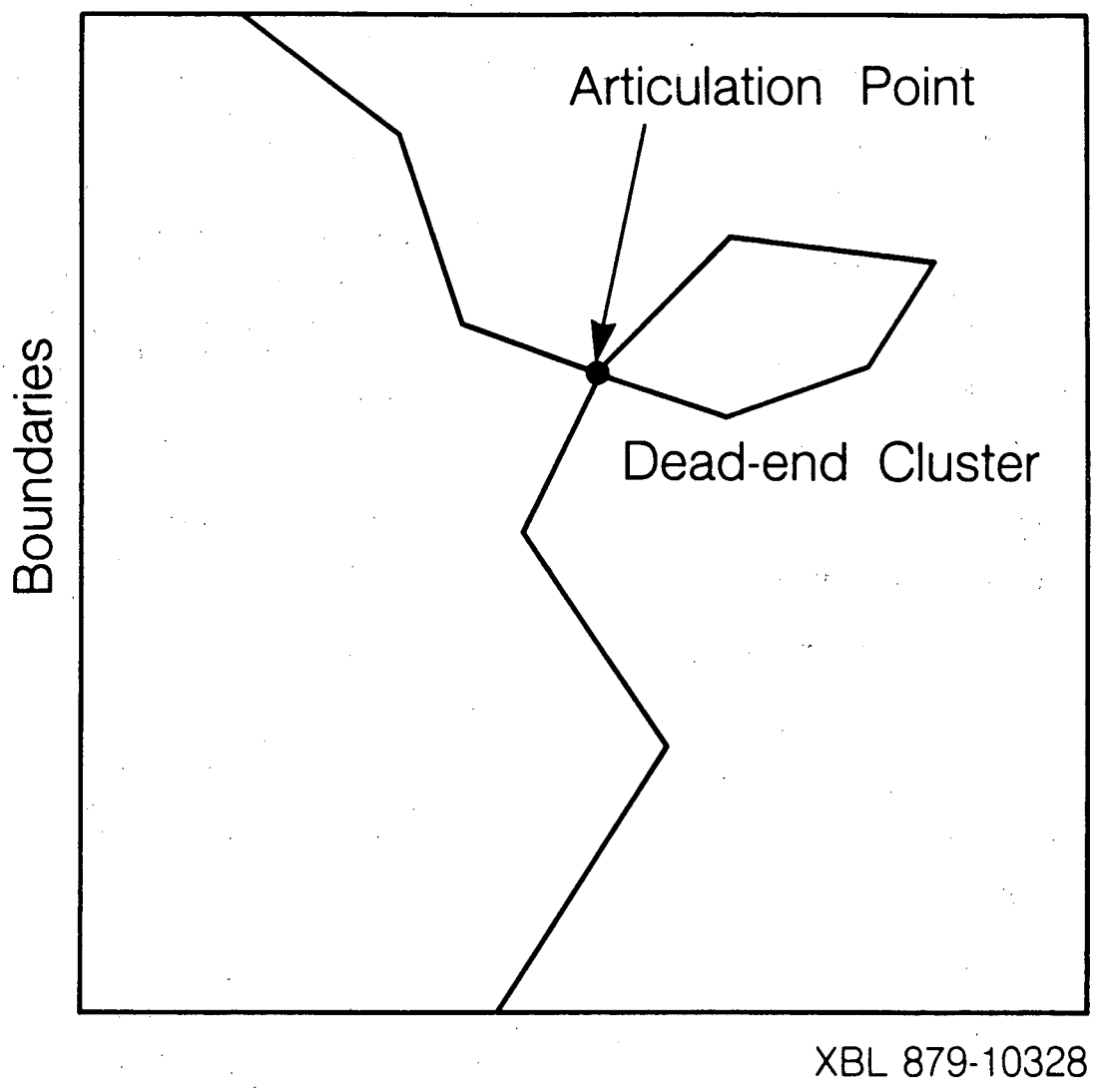


Figure 3.5. Second sufficient condition. Respective positions of boundaries, articulation points and dead-end cluster.

We must pass through its articulation point to reach the dead-end from the boundary. Therefore the nodes in the dead-end have higher numbers than the articulation point. Reversing this proposition, we find that when we jump from one node to another along a line element, if the node number decreases, we cannot be entering a dead-end cluster. The second sufficient condition is therefore, that a node is active if it is connected by a single line element to an active node with a number higher than itself.

We use the above sufficient condition to flag active paths, that is paths in which all the nodes are active. Starting in sequence from each of the nodes flagged as active during the previous search, a restricted breadth-first search is performed. When studying a given node, instead of recording all the nodes connected to it by a line element, we select only those with a lower number. Because we started from an active node, all the nodes we go through are active and we flag them as such.

We may not be flagging all the active nodes in the mesh, since the condition we are using is sufficient but not necessary. For example, in Figure 3.4, nodes 8, 11, 12, 15, 16 will not be flagged. We thus will need to study again the part of the mesh which has not been flagged as active yet. To initialize this next forward search we will need to know where there are branches we did not select in the restricted search. So during this downward search, we make as initial node for the next search any node we encountered that is connected to a node not flagged active and with a higher number. Figure 3.6 shows the example mesh at the end of the first downward search. Node 9 qualified to be an initial point for the next forward search because it is connected to mode 12. Node 6 qualified also because it is connected to Node 8.

Starting from the source nodes defined during the downward search, we perform the same forward search as before, but avoiding all nodes already known to be active. We renumber the nodes we encounter in sequence, assign to them new sources, and flag more active nodes in the same manner as before. The downward search is then performed starting from the newly found active nodes only, and the process is repeated until all active nodes have been flagged (Figure 3.7).

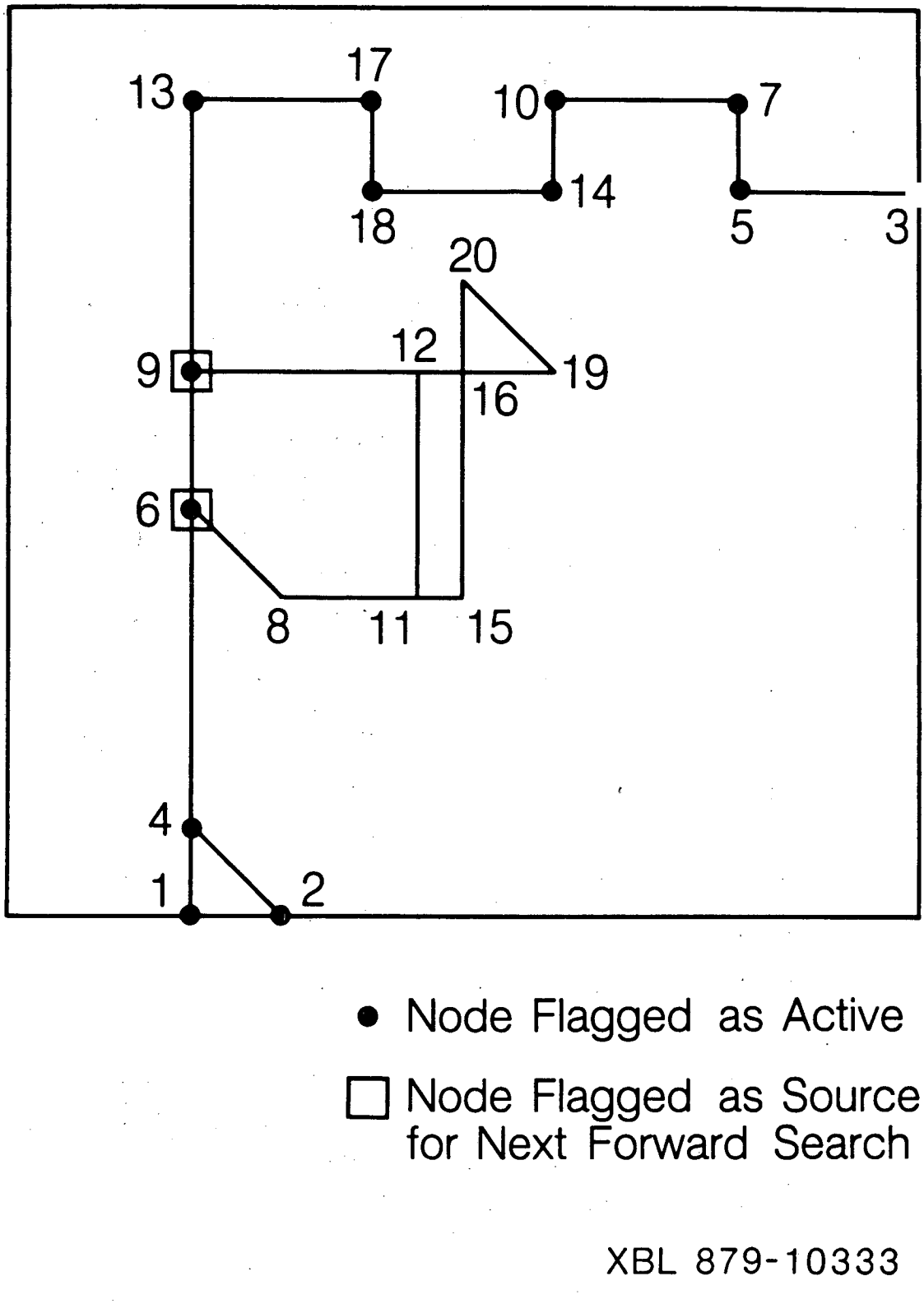


Figure 3.6. Mesh from Figure 3.3 at the end of the first downward search.

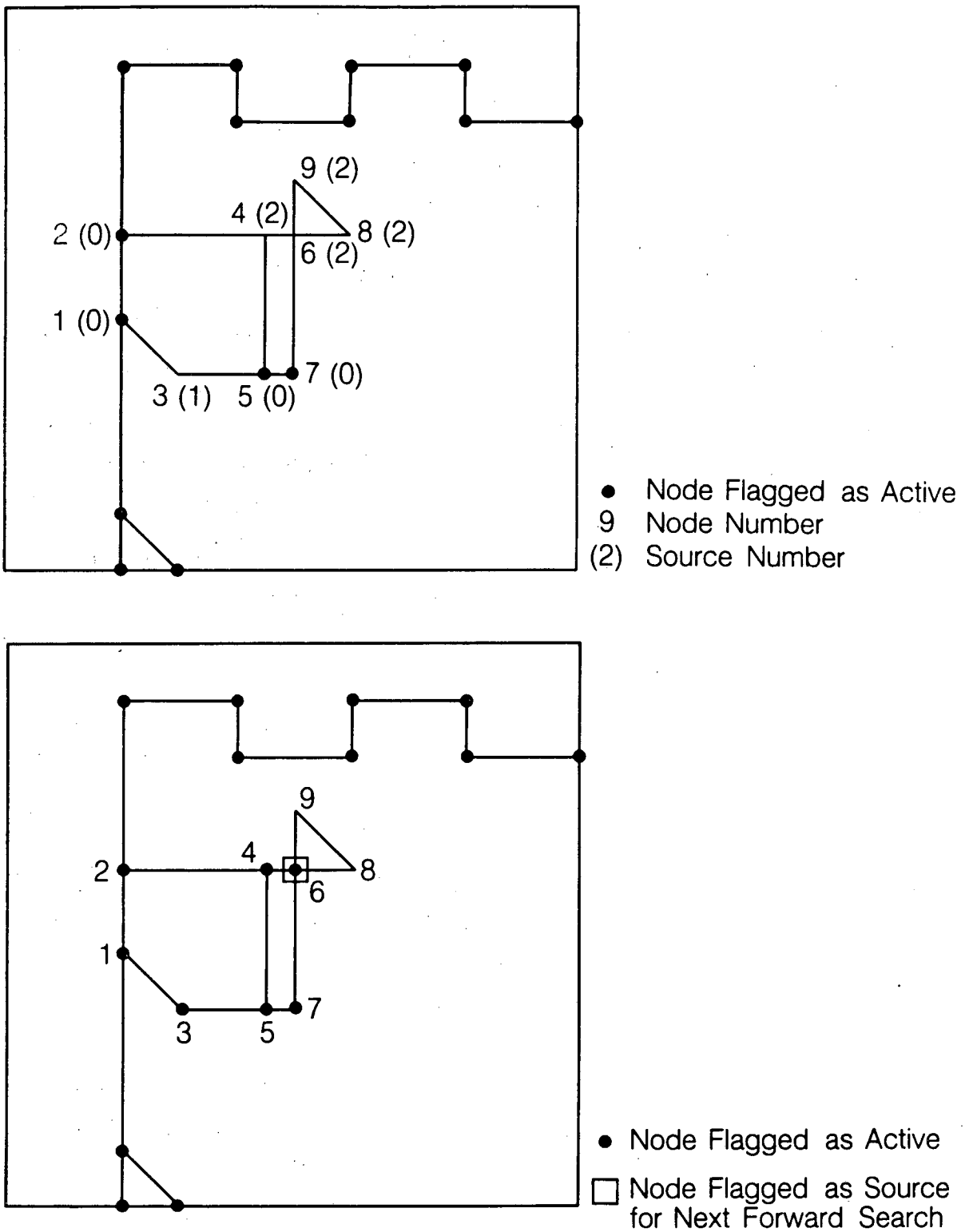


Figure 3.7. Mesh from Figure 3.3, second loop of searches. (a) forward search, (b) downward search: only one source is left for the next forward search. The algorithm stops, nodes 8 and 9 are discarded.

XBL 879-10329

The process is stopped after a downward search if not more than one new source node has been flagged. After a forward search it is stopped if no new active node has been flagged.

3.3 Node Renumbering and Output

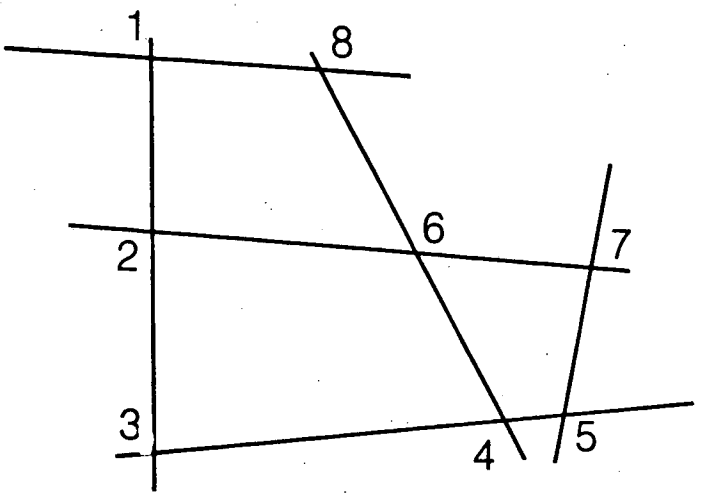
3.3.1 Banded Matrices

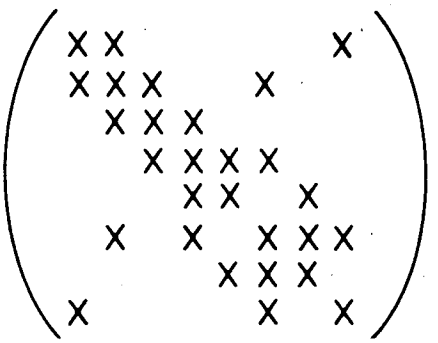
The bandwidth of a matrix is the maximum of the difference between the numbers of two nodes in the same row, plus one. For a symmetric matrix, only half of the matrix is generally stored, either the upper or the lower triangle bounded by the diagonal. So the half-bandwidth is defined for a symmetric matrix as the maximum distance along any row between the diagonal and a non zero off diagonal term, plus one. Numerical algorithms for the solution of linear systems take advantage of a narrow bandwidth in various ways. The most common way is to store only a number of terms equal to the bandwidth for each row of the matrix, thus reducing the storage requirements and the number of operations needed. If either a complete or an incomplete Choleski decomposition is performed, only terms within the band are filled in (i.e., are changed from zero to non zero). Thus the narrow bandwidth also drastically reduces the storage and computer time requirements. These solving procedures are discussed in more detail in paragraph 4.2.

3.3.2 The Cuthill-McKee Algorithm

The algorithm used to minimize the bandwidth was published by Cuthill and McKee (1969). It consists of searching the network level by level as explained in Section 3.2 and renumbering the nodes as they are reached. For example, in Figure 3.8a if node one is put in level one, then level two consists of nodes (2 and 8), level 3 of nodes (3 and 6), and so on, and the renumbering yields the node numbering shown in Figure 3.8b. The matrix obtained has a half bandwidth of 4 as compared to the matrix in Figure 3.8a which has a half bandwidth of 8.

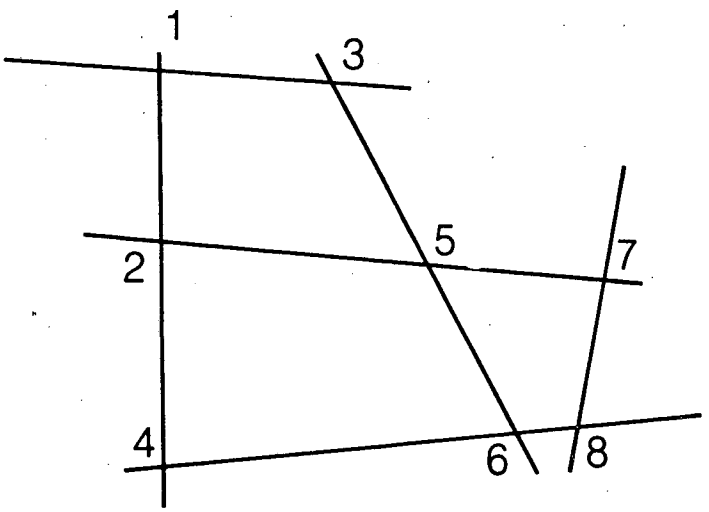
The search is initialized by putting all boundary nodes on side one in level one. Once all nodes connected to side one have been screened, all boundary nodes on side two which have

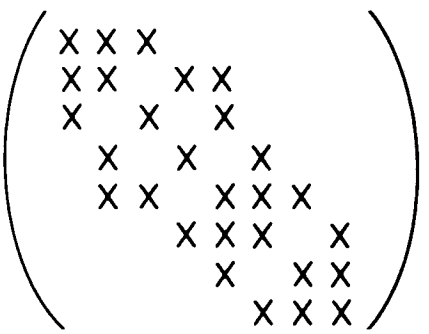




Half-Bandwidth 8

a) Old Numbering and Matrix





Half-Bandwidth 4

b) New Numbering and Matrix

XBL 882-10055

Figure 3.8. Node renumbering, after Robinson (1982).

not yet been screened are put in the next level and the search is reinitialized. This goes on until all sides have been exhausted (four sides in 2 dimensions or 6 sides in 3 dimensions).

Subroutine PLOT then writes plotting files, linesnn.dat, with nn = 01, 02, 03 etc., for each flow region studied. Then, the bandwidth is computed by PROUT, and this subroutine prints the finite element network for use by programs LINEL (Chapter 4) or TRINET (Karasaki, 1987).

4.0 PROGRAM LINEL

LINEL computes the steady state flux through a mesh of line elements previously processed by program RENUM. LINEL was first developed by Wilson (1970).

For a given set of boundary conditions, the head at each node and the flux in each element are computed. The output of the program consists of the sum of the fluxes through each side of the flow region.

The program can deal with two dimensional meshes generated by FMG or with three dimensional meshes generated by FMG3D-CHANGE (Gilmour et al., 1986; Billaux and Long, 1988). For two dimensional problems, study regions can be defined as an option. These are smaller rectangular regions centered in the flow region. Whenever such regions are specified, the flux through their boundaries and the average gradient inside them are also computed.

4.1 Building the Linear System of Equations

Using Darcy's law for each element and writing the conservation of mass at each node of the network yield the system of linear equations that needs to be solved given the boundary conditions.

4.1.1 Governing Equations

In each line element, Darcy's law states that the flux through the element is proportional to the gradient of charge in the element, the proportionality constant being the transmissivity of the element. So given an element of length l_{ij} and transmissivity t_{ij} joining to nodes i and j, and given the charges h_i and h_j at nodes i and j, the flux Q_{ij} from i to j is given by:

$$Q_{ij} = t_{ij} \frac{(h_i - h_j)}{l_{ij}} \tag{4.1}$$

At each internal node there is no creation of mass, except if the node is specified as a source.

So for all i's, we have

$$\sum_j Q_{ji} = q_i \tag{4-2}$$

where q_i is the imposed flux at node i, which is zero for most or all internal nodes. The sum is over all nodes j connected to node i by one line element. Note that this equation cannot be written for imposed-head nodes, where some mass is effectively entering or leaving the mesh. At these nodes, we simply write the identity between the head and the imposed head.

4.1.2 Linear System

The linear system is obtained by simply writing Equation 4-2 for all nodes, with the Q_{ij}'s written in terms of the k_i's using Equation 4-1. If the mesh has n nodes, the matrix will then be of order n.

Each line i of the linear system has the form:

$$\sum_j \frac{t_{ij}}{l_{ij}} h_i - \sum_j \frac{t_{ij}}{l_{ij}} h_j = q_i \tag{4-3}$$

where the summation on j's is the same as in Equation 4.2.

If this system of equations is written:

$$A\,h = b$$

then from Equation 4.3, the diagonal terms in A are

$$A_{ii} = \sum_j \frac{t_{ij}}{l_{ij}}$$

the off diagonal terms are:

$$A_{ij} = -\frac{t_{ij}}{l_{ij}}$$

if an element ij exists; otherwise,

$$A_{ij} = 0$$

and the b terms are:

$$b_i = q_i$$

The above terms are complete if there are no imposed head boundary nodes. If a node k has its head imposed, the line k of the system reduces to

$$\left[\sum_j \frac{t_{kj}}{l_{kj}}\right] h_k = \left[\sum_j \frac{t_{kj}}{l_{kj}}\right] H_k \tag{4-4}$$

where H_k is the imposed head.

We can then rewrite Equation 4-3 separating the nodes into non-imposed head nodes and imposed head nodes:

$$\sum_j \frac{t_{ij}}{l_{ij}} h_i + \sum_k \frac{t_{ik}}{l_{ik}} h_i - \sum_j \frac{t_{ij}}{l_{ij}} h_j - \sum_k \frac{t_{ik}}{l_{ik}} H_k = q_i \tag{4-5}$$

where node i is a non-imposed head node, the sum of j's is over all non-imposed head nodes connected to node i by a line element, the sum in k's is over all imposed-head nodes connected to node i by a line element. Passing the fourth term in the left hand side (known term) to the right, we obtain:

$$\sum_j \frac{t_{ij}}{l_{ij}} h_i + \sum_k \frac{t_{ik}}{l_{ik}} h_i - \sum_j \frac{t_{ij}}{l_{ij}} h_j = q_i + \sum_k \frac{t_{ik}}{l_{ik}} H_k \tag{4-5}$$

The left hand side represents matrix A, and the right hand side is the b vector. So we obtain the final expression for the terms of A and b:

- diagonal terms in A:

$$A_{ii} = \sum_j \frac{t_{ij}}{l_{ij}} \tag{4-6}$$

  where the sum in j's is over all nodes j, either imposed head or not, connected to node i by a line element

- off diagonal terms in A:

  if an element ij exists and neither node i nor node j is an imposed-head node, then

$$A_{ij} = -\frac{t_{ij}}{l_{ij}} \tag{4-7}$$

  otherwise

$$A_{ij} = 0 \tag{4-8}$$

- terms in b:

  if node i is a non-imposed head node

$$b_i = q_i + \sum_k \frac{t_{ik}}{l_{ik}} \tag{4-9}$$

  where the sum in k's is over all imposed-head nodes connected to node i by a line element.

  If node k is an imposed head node,

$$b_k = \sum_j \frac{t_{kj}}{l_{kj}} \tag{4-10}$$

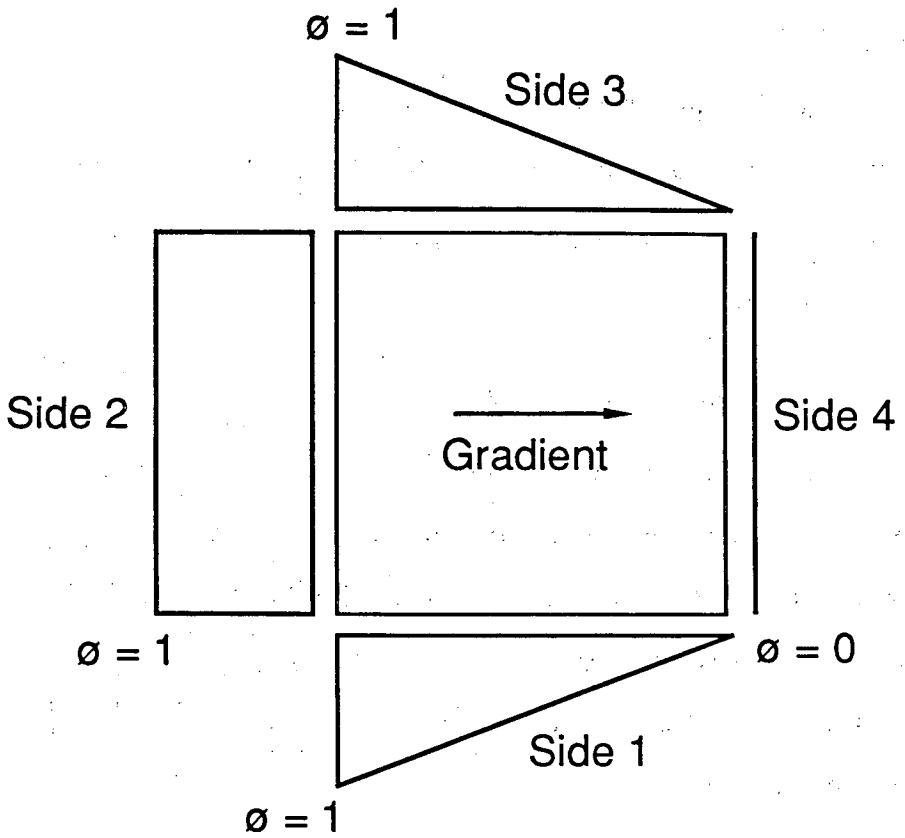
  where the sum in j's is over all nodes connected to node k by a line element.

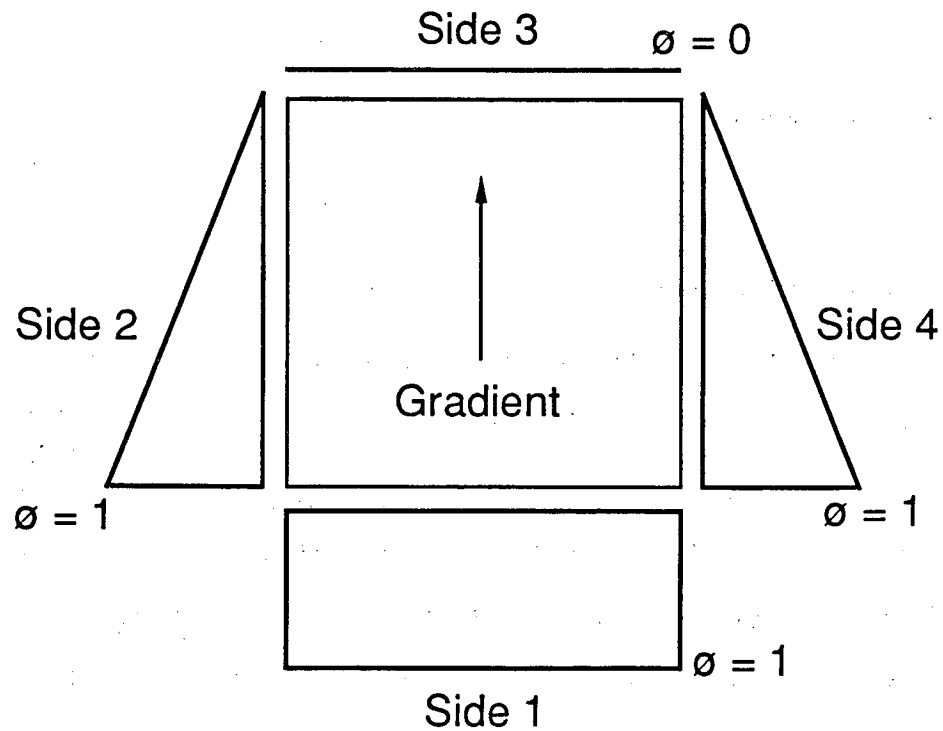
4.1.3 Implementation

Subroutine CPHI builds A and b. A loop is performed over all the line elements in the network. For each of them, the numbers of the nodes at the extremities are retrieved, the right terms in the diagonal of A are added, then depending on the status of the two nodes (either imposed head or not), the proper terms are added into b or put off the diagonal into A. Note that several (up to three) b vectors can be built at the same time, to compute, for example, flow in a given mesh under gradients in several different directions.

In order to save computing time when solving the linear system, when filling matrix A and vector b, the lines of A and b corresponding to imposed-head nodes are switched to the bottom of the tables. Then the order of the matrix passed to the linear solver will be only the number of nodes where head is not imposed. In this way, the program avoids solving the lines of the system where it reduces to an identity.

For two-dimensional networks, the default number of boundary conditions is two. The boundary conditions used are: 1) the conditions imposed by the users, and 2) the same conditions rotated 90° (Figure 4.1). In this way, two different directional permeabilities are found at the same time in the process of computing a porous medium equivalent tensor (see Chapter 5). This default setup is overridden whenever an imposed flux boundary node is present in the





XBL 882-10062

Figure 4.1. Two sets of boundary conditions for directional permeability.

mesh.

For three-dimensional networks, the number of differing sets of boundary conditions is input by the user, and each set of boundary conditions is fully defined by the user.

The way matrix A is stored in the computer depends on the solver used later. The solver requires A to be stored in banded fashion.

4.2 Solving the Linear System

A system of equations representing flow in a line element network has several interesting properties. The matrix A is symmetric positive definite, and is also generally sparse with a narrow bandwidth. The symmetry follows directly from the expression for the off-diagonal terms in A. The fact that A is both positive and definite stems from the relationship between the off-diagonal and diagonal terms in every line. From the construction of A (Equations 4-6, 4-7 and 4-8), it follows that:

$$A_{ij} < 0$$

for all i and for all j such that element ij exists.

$$A_{ii} = -\sum_{j \neq i} A_{ij}$$

if no node j is an imposed head node;

$$A_{ii} > -\sum_{j \neq i} a_{ij}$$

if at least one node j is an imposed-head node.

Simple algebra then can prove a sufficient condition for positive definiteness which is that for any vector x,

$$x^t A\, x > 0.$$

Recall that an off-diagonal element of matrix A, A_{ij} is non-zero only if a line element joins the two nodes i and j. This implies that on a given line i of A, there will be as many off-diagonal non zeros as there are line elements having node i as an endpoint. Experience shows that almost all nodes are the endpoints of two to four line elements. A node can be the endpoint of

more than four elements in two instances only. 1) If three fractures intersect at the same point or at almost the same point, program RENUM merges the several resulting nodes into one. 2) All the nodes intersecting a given imposed flux hole (i.e., a well) are also shrunk into one by RENUM. So the number of non-zeros on any given line of A averages less than five (four off-diagonals plus one diagonal). Considering that networks of 10,000 nodes are not uncommon, the matrix A for such networks is made of 99.95% zeros and 0.05% non zeros. In fact, the number of non-zero off-diagonals we need to store can be cut in half because of the symmetry.

The narrow bandwidth of the system is discussed in Chapter 2. Note that even for a sparse system, renumbering the nodes to minimize the bandwidth is efficient and reduces overall computer time.

The solver SYMSOL is a fortran subroutine that solves the linear system of equations with A stored in lower triangular banded form. SYMSOL uses the lower triangle decomposition method to solve the system of equations. This solver is implemented in FORTRAN-77, so it is easily portable to any computer.

4.3 Computing Fluxes

The solution of the linear system is an array containing the hydraulic head at each node in the network. The flux entering the system at each imposed head boundary node is computed. The sum of the fluxes through each side of the flow region is then computed and printed. If required, the head and flux at the boundaries of a smaller study region are also computed and printed.

4.3.1 Flow Region

Once the head at each node is known, it is a simple matter to compute the flux through any element of the network, by using Equation 4-1. Subroutine CUNK loops over all the elements. When an endpoint i of an element is found to be an imposed-head node, the flux leaving node i through the element is computed.

Subroutine PINFO then prints the header for the output file, and optionally all the node and element characteristics together with heads at nodes and velocities in elements.

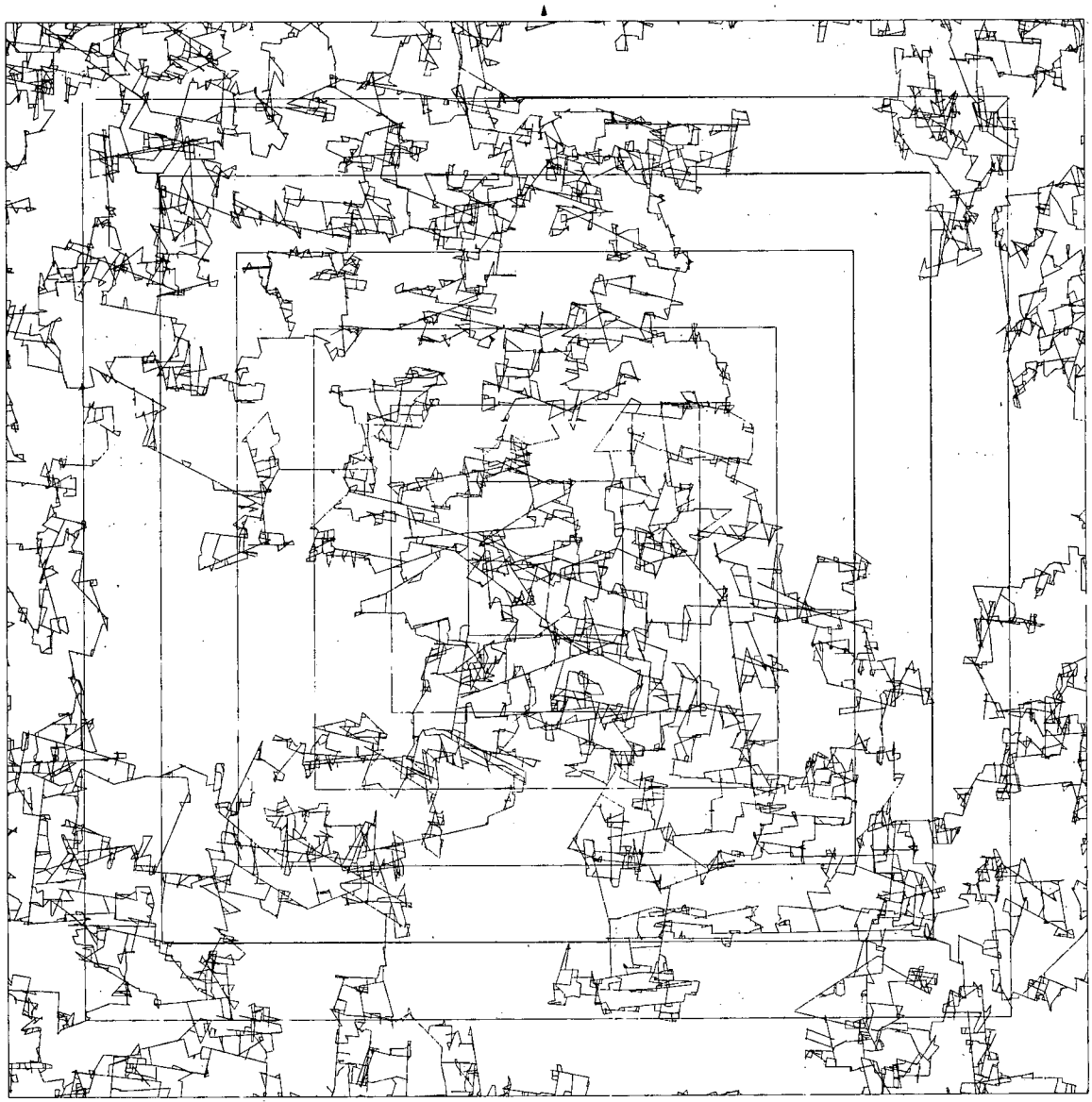
The fluxes leaving all the nodes through each boundary side are summed up by subroutine SFLUX. These fluxes are output, as well as the inverse of their square root for two-dimensional cases, to be used by program ELLFMG (Chapter 5).

Also, if flux has been imposed at any node, the head at this node is printed. The average number of fractures per unit length (2-D) or area (3-D) intersecting each side of the flow region is also computed and printed.

4.3.2 Study Regions

For computing equivalent porous medium permeabilities in the two-dimensional case, the boundary conditions that are imposed (see Section 2.3.3) are likely to cause a consistent overestimation of the permeability, because flow may be forced through fractures or fracture clusters that only transect the corner of the region. The importance of this effect decreases as the scale of measurement is increased. This problem is handled by using "study regions" (Figure 4.2).

A study region is centrally located inside the flow region, and a border region of width Δ is left between the study region and the boundaries of the flow region. The boundary conditions are still applied to the flow region sides, but fluxes are examined at the boundaries of the study region. For a large enough Δ, the border effect is all but eliminated. But another difficulty then arises in defining the gradient J over the study region. The gradient over the flow region is clearly defined by the applied boundary conditions. The gradient J_s over the study region, however, can be defined in one of two ways. We can take J_s equal to J operating on the flow region, which we call the "global gradient". Alternatively we can calculate a local gradient. We do this by computing the head at the intersection between any fracture and the boundaries of the study region. By taking the difference between the average heads on the outflow and inflow sides, we are able to find the gradient which would theoretically be meas-



XBL 863-1170

Figure 4.2. A 70 m by 70 m flow region with six nested study regions.

ured in situ by monitoring heads at the boundary of the region. Note that the global and local gradients should be identical if the medium behaves as a continuum.

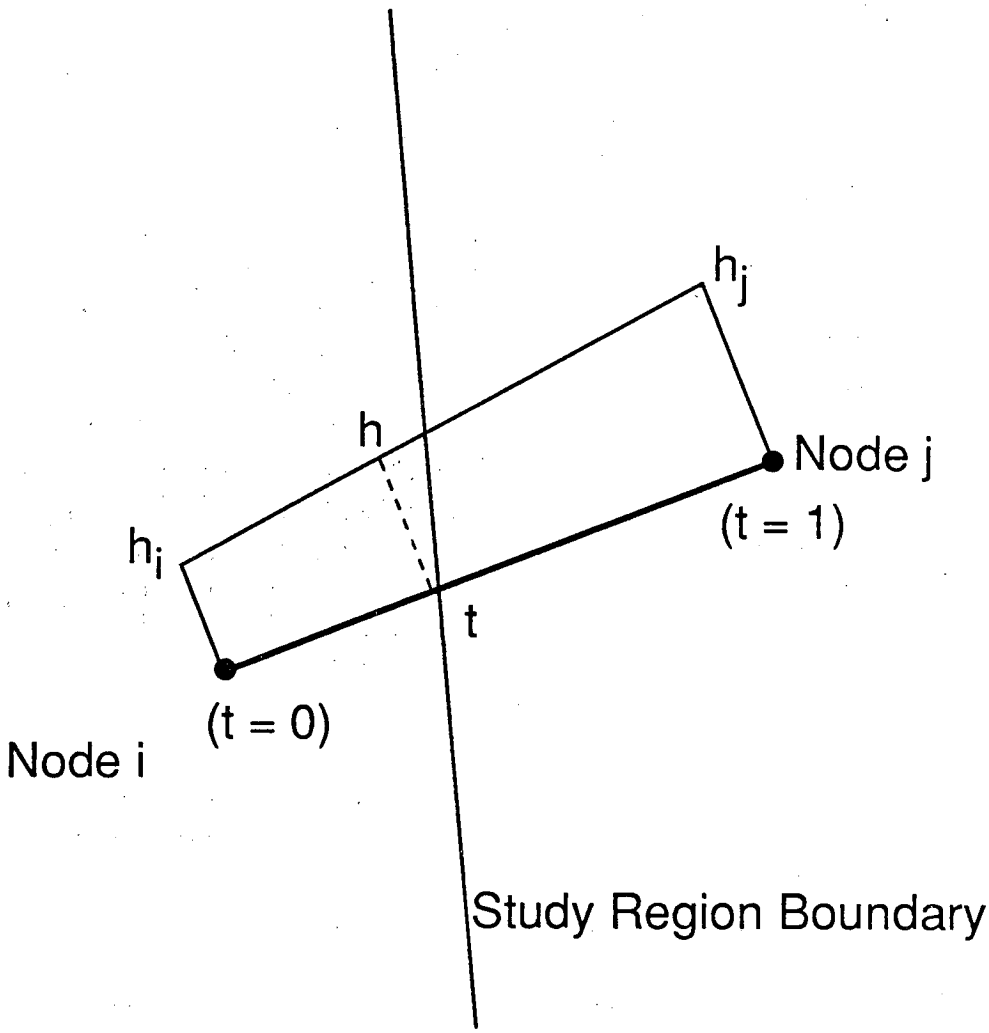
Subroutine STUDY optionally performs study region computations if such regions have been specified. The intersection points between fractures and study region boundaries are determined. The head and flux at these points are computed using the known heads at all the nodes in the network. The appropriate sums of fluxes and averages of heads are then computed, and permeabilities computed using local or global gradients are output, as chosen by the user. Note that several (up to 20) study regions can be specified for the same run.

The intersections between each study region and the elements of the network are found using part of the algorithm described in Section 1.2.2 for truncating fractures lying partially outside the flow region. When an intersection is found, the flux in the element is computed using Equation 4-1. The head at the intersection is also computed using the assumption that the head varies linearly along any element. If the endpoints of the element are nodes i and j, and the intersection is at a relative distance t from node i (Figure 4.3), with t = 0 at node i and t = 1 at node j, then:

$$h = h_i + t(h_j - h_i) \tag{4-11}$$

where h is the head at the intersection, h_i is the known head at node i and h_j is the known head at node j. Then the head and flux are added to the sum of heads and the sum of fluxes stored for the particular side of the study region. The squares of the heads are also added up for each side in order to compute the standard deviation of heads.

Once all the elements have been screened, the average and standard deviation of heads on each side of all study regions are computed and printed. If needed the local gradient is computed, as the ratio between the difference in average heads from the inflow to the outflow side, and the size of the study region. Using the sum of the fluxes on the outflow side and inflow side, and either the global gradient or the local gradient, the average permeability is computed and printed in a file to be used as input for program ELLFMG.



XBL 882-10063

Figure 4.3. Notations for computing the head at study region boundaries.

5.0 PROGRAM ELLFMG

As mentioned before, the program ELLFMG is one of the links in a chain of programs, which consists of FMG, RENUM, LINEL, ELLFMG, DIMES and ELLP, built to analyze fluid flow through a two-dimensional fracture network. The input to program ELLFMG is a list of directional permeabilities obtained by LINEL. This list is built using the ability of the programs FMG, RENUM and LINEL to process several flow regions in one run, for one given set of generated fractures. It can also be built by appending results from different runs, making sure that the same pseudo-random generation process is repeated for each run. An example of several flow regions for the same mesh is shown in Figure 5.1. For each of them, flow is computed under a gradient from side 2 to side 4. Note that since LINEL is able to compute flow under both a gradient from side 2 to side 4 and a gradient from side 3 to side 1 at the same time (Chapter 4), Figure 5.1g (90° rotation) corresponds to solving the same matrix as Figure 5.1a (0° rotation). Identically, pairs of rotations (15° and 105°), (30° and 120°), (45° and 135°), (60° and 150°), (75° and 165°), each correspond to solving one linear system. From the obtained values for the directional permeability in several direction, $K_g(\alpha)$, ELLFMG determines the three components of the permeability tensor, K_{ij}, which fits best these results. Then the principal values (eigen values) and principal axes (eigen vectors) of the permeability tensor are computed. ELLFMG also produces a quantitative measure of the difference between the measured values, $K_g(\alpha)$, and the best-fit values.

5.1 Permeability Ellipse and Permeability Distributions

For an ideal anisotropic homogeneous porous medium the directional permeability, K_g, measured in the direction of the gradient (α), is defined by the following equation:

$$q_i n_i = K_g J \tag{5-1}$$

where n_i is a unit vector in the direction of the gradient, J is the magnitude of the gradient, and

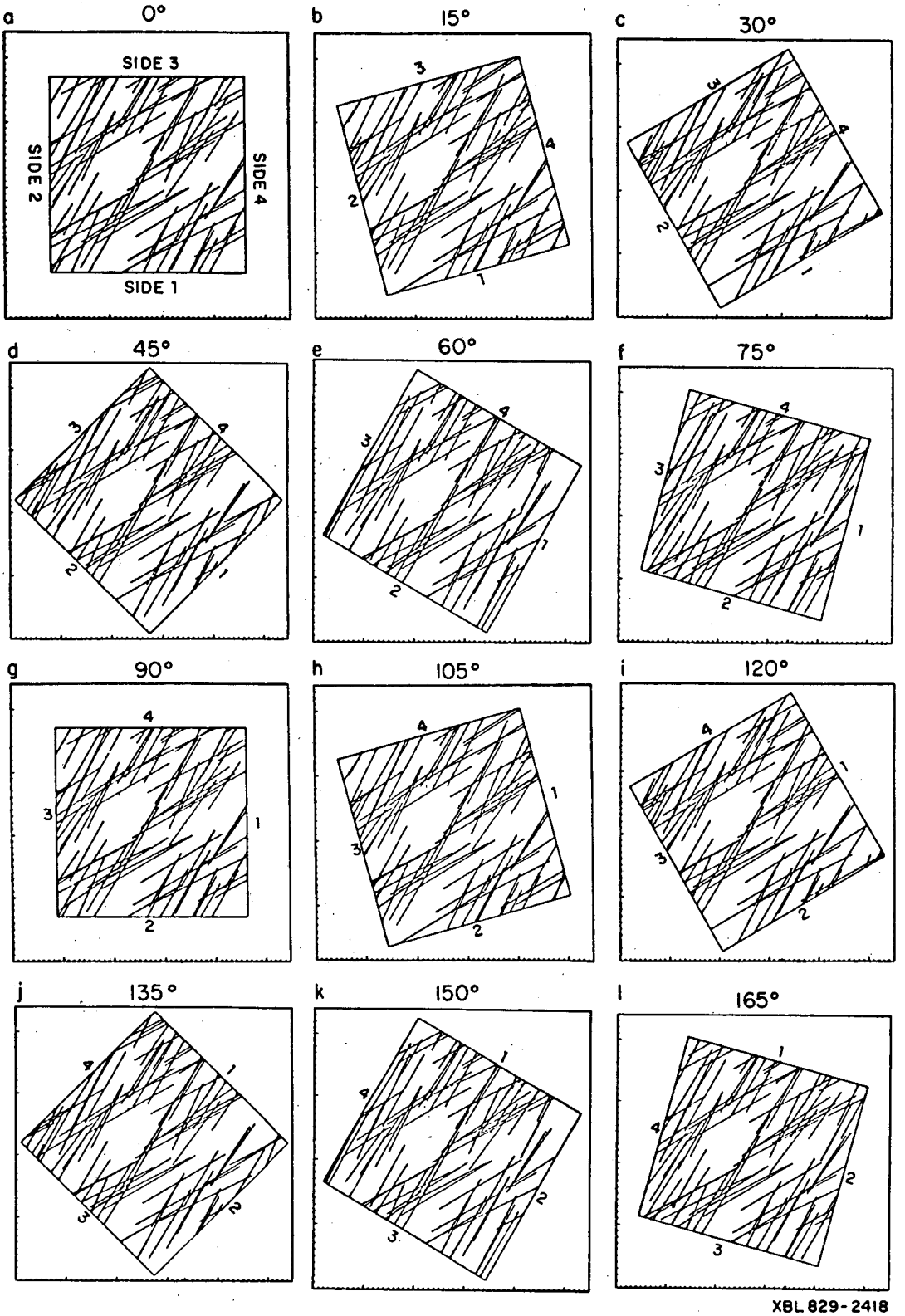


Figure 5.1. Flow regions with various orientations for directional permeability studies.

q_i is the specific flux. Solving Darcy's law for q_i and substituting this into 5-1 gives

$$K_{ij}J_jn_i = K_gJ, \tag{5-2}$$

and since $J_j/J = n_j$ we have

$$K_g = K_{ij}n_in_j, \tag{5-3}$$

or

$$K_g = K_{11}\cos^2\alpha + 2K_{12}\cos\alpha\sin\alpha + K_{22}\sin^2\alpha, \tag{5-4}$$

where n_1 and n_2 are direction cosines and $n_1 = \cos\alpha$, $n_2 = \sin\alpha$.

If $1/\sqrt{K_g}$ is plotted in the direction α (the direction of the gradient), then $n_1 = \cos\alpha = x\sqrt{K_g}$ and $n_2 = \sin\alpha = y\sqrt{K_g}$. Equation 5-4 becomes

$$K_g = K_{11}x^2K_g + 2K_{12}xyK_g + K_{22}y^2K_g, \tag{5-5}$$

$$1 = K_{11}x^2 + 2K_{12}xy + K_{22}y^2, \tag{5-6}$$

$$1 = K_{ij}x_ix_j \qquad \text{where } x_i = \begin{Bmatrix} x \\ y \end{Bmatrix}. \tag{5-7}$$

Equation 5-7 is the quadratic form of the equation of an ellipse, which we will call the permeability ellipse. This ellipse has semi-axes of length $1/\sqrt{K_1}$ and $1/\sqrt{K_2}$ which correspond to the principal axes of the permeability tensor. Note that the major axis of the ellipse is in the direction of minimum permeability.

If each measurement of $K_g(\alpha)$ can be considered an independent measurement of the value of K_{ij}, then methods of statistics can be used to estimate the parameters K_{11}, K_{12} and K_{22}. The statistical technique can be used on measurements of $K_g(\alpha)$ from different, equally incremented directions on one fracture pattern or the combined measurements from any number of fracture pattern realizations.

Distribution of $K_g(\alpha)$

In a random fracture pattern, the measured values of $K_g(\alpha)$ will not all plot in a single, unique ellipse (Figure 5.2). In order to use all of the individual measurements to derive a single, most representative set of parameters for the permeability tensor, we must assume that each measurement is independent and similarly distributed. Figure 5.2 shows an example of a

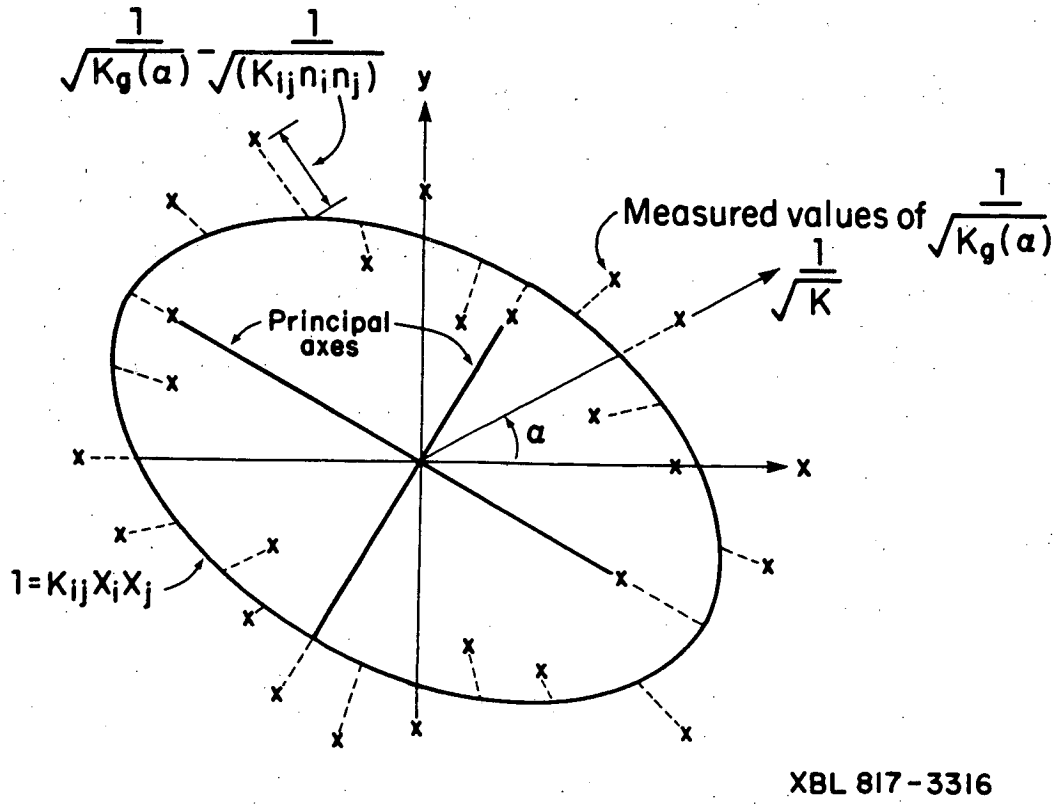


Figure 5.2. A set of directional permeability measurements plotted as $1/\sqrt{K_g}$ in polar coordinates.

set of measurements, $K_g(\alpha)$, and a possible ellipse with parameters K_{11}, K_{12} and K_{22}. Each measurement can be assumed to be distributed about a different mean which is a point on the ellipse determined by α. Therefore, the value of the mean for each measurement depends on α. Thus, each $K_g(\alpha)$ is considered to be distributed with the same form but each has a different or shifted mean. The variance of each $K_g(\alpha)$ is assumed to be identical. In this way, all the measurements are considered as one population.

It would be very useful to be able to define a likely distribution function for $K_g(\alpha)$, but this is not easily done. The normal distribution does not match the data because $K_g(\alpha)$ can never be less than zero. A lognormal distribution is not proper because the probability of $K_g(\alpha) = 0$ is finite, not zero. Exponential, Gamma and Beta distributions also are not suitable. A normal distribution truncated at $K_g(\alpha) = 0$ is a likely choice. Unfortunately, assuming this distribution leads to a contradiction with the basic assumption that all the measurements are members of the same distribution. At each angle α, the mean value of the distribution is different. However, since all the distributions are truncated at zero, the difference between the mean value and the truncation limit is different for each value of α. This means that each measurement must be a member of a different, truncated normal distribution and not just a shifted one as required in the original assumptions. Since a simple, likely, distribution form for $K_g(\alpha)$ which conforms to the basic assumptions cannot be identified, a least squares regression technique is used to derive estimates of the parameters K_{11}, K_{12} and K_{22}.

5.2 Finding the Permeability Parameters

The regression technique used is based on a technique, discussed by Scheidegger (1954), which will also be briefly described.

In order to find the best bit ellipse, we minimize the function R :

$$R = \sum_{n=1}^{N} \left[K_g(\alpha_n) - (K_{ij} n_i n_i) \right]^2. \tag{5-8}$$

In fact we are not directly regressing to the best-fit ellipse. We are trying to find the parameters K_{11}, K_{12}, and K_{22} which best fit the data expressed by $K_g(\alpha_n)$ in Eq. 5-5. Figure 5.3

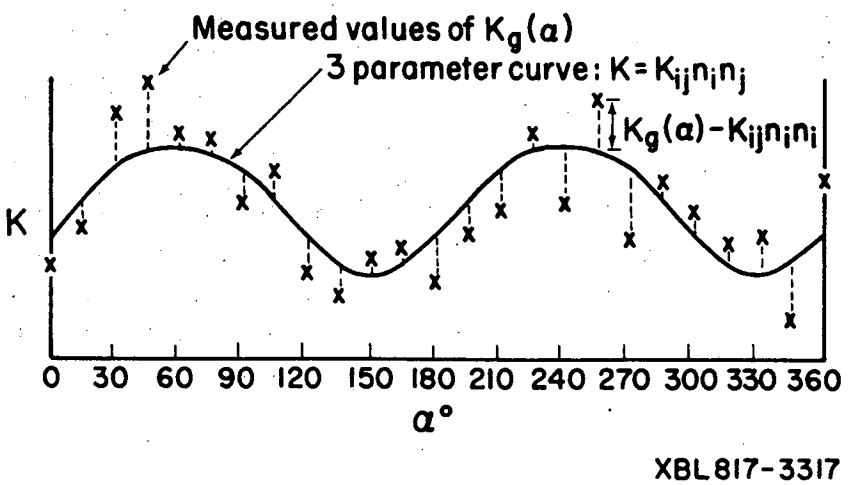


Figure 5.3. A set of directional permeability measurements plotted in cartesian coordinates.

illustrates the type of three parameter curve which is fitted to the data.

A similar technique was used by Scheidegger (1954). Scheidegger minimized the function

$$R = \sum_{m=1}^{N} \left[(K_f^1(\alpha_m))^{-1} - (K_{ij})^{-1} m_i m_j \right]^2. \tag{5-9}$$

Although not stated by Scheidegger, this regression technique applies to measurements of permeability $K_f(\alpha)$ made in the direction of flow. Thus, m_i is a unit vector in the direction of flow. To see this, note that permeability in the direction of flow is defined by

$$\frac{1}{K_f} = \frac{J_i m_i}{q}, \tag{5-10}$$

where q is the specific flux and J_i is the gradient.

Substituting Darcy's law we have

$$\frac{1}{K_f} = q_i (K_{ij})^{-1} \frac{m_i q_j}{q}, \tag{5-11}$$

$$\frac{1}{K_f} = (K_{ij})^{-1} m_j m_i. \tag{5-12}$$

So Equation 5-8 is effectively the same as Equation 5-8, except that in 5-9, K_{ij} becomes the inverse of the permeability tensor.

The solution of the regression equations is the same as the solution given by Scheidegger (1954). The equations are

$$\frac{\partial R}{\partial K_{11}} = 0 = \sum_{n=1}^{N} -2 \left[K_g(\alpha_n) - K_{ij} n_i n_i \right] \cos^2\alpha_n \quad , \tag{5-13}$$

$$\frac{\partial R}{\partial K_{12}} = 0 = \sum_{n=1}^{N} -4 \left[K_g(\alpha_n) - K_{ij} n_i n_i \right] \cos\alpha_n \sin\alpha_n \quad , \tag{5-14}$$

$$\frac{\partial R_{II}}{\partial K_{22}} = 0 = \sum_{n=1}^{N} -2 \left[K_g(\alpha_n) - K_{ij} n_i n_i \right] \sin^2\alpha_n. \tag{5-15}$$

Rearranging, expanding $K_{ij} n_i n_j$, and putting in matrix form we have

$$\begin{bmatrix} \sum_{n=1}^{N} \cos^4\alpha_n & \sum_{n=1}^{N} 2\cos^3\alpha_n\sin\alpha_n & \sum_{n=1}^{N} \sin^2\alpha_n\cos^2\alpha_n \\ \sum_{n=1}^{N} \cos 3\alpha_n\sin\alpha_n & \sum_{n=1}^{N} 2\sin^2\alpha_n\cos^2\alpha_n & \sum_{n=1}^{N} \sin^3\alpha_n\cos\alpha_n \\ \sum_{n=1}^{N} \cos^2\alpha_n\sin^2\alpha_n & \sum_{n=1}^{N} 2\sin^3\alpha_n\cos\alpha_n & \sum_{n=1}^{N} \sin^2\alpha_n \end{bmatrix} \begin{bmatrix} K_{11} \\ K_{12} \\ K_{22} \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{n=1}^{N} K_g(\alpha_n)\cos^2\alpha n \\ \sum_{n=1}^{N} K_g(\alpha_n)\cos\alpha_n\sin\alpha_n \\ \sum_{n=1}^{N} K_g(\alpha_n)\sin^2\alpha_n \end{bmatrix} \tag{5-16}$$

Now, if for each fracture mesh, measurements are made at equal angle intervals from 0 to 2π, all sums with only odd powers of sine and cosine drop out and the equation becomes:

$$\begin{bmatrix} \sum_{n=1}^{N} \cos^4\alpha_n & 0 & \sum_{n=1}^{N} \sin^2\alpha_n\cos^2\alpha_n \\ 0 & \sum_{n=1}^{N} 2\sin^2\alpha_n\cos^2\alpha_n & 0 \\ \sum_{n=1}^{N} \cos^2\alpha_n & 0 & \sum_{n=1}^{N} \sin^4\alpha_n \end{bmatrix} \begin{bmatrix} K_{11} \\ K_{12} \\ K_{22} \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{n=1}^{N} K_g(\alpha_n)\cos^2\alpha_n \\ \sum_{n=1}^{N} K_g(\alpha_n)\cos\alpha_n\sin\alpha_n \\ \sum_{n=1}^{N} K_g(\alpha_n)\sin^2\alpha_n \end{bmatrix} \tag{5-17}$$

Solving for K_{11}, K_{12}, and K_{22} gives

$$K_{11} = \frac{\left[\sum_{n=1}^{N} K_g(\alpha_n)\sin^2\alpha_n\right]\left[\sum_{n=1}^{N} \cos^2\alpha_n\sin^2\alpha_n\right] - \left[\sum_{n=1}^{N} K_g(\alpha_n)\cos^2\alpha_n\right]\left[\sum_{n=1}^{N} \sin^4\alpha_n\right]}{\left[\sum_{n=1}^{N} \sin^2\alpha_n\cos^2\alpha_n\right]^2 + \left[\sum_{n=1}^{N} \cos^4\alpha_n\right]\left[\sum_{n=1}^{N} \sin^4\alpha_n\right]} \tag{5-18}$$

$$K_{12} = \frac{\sum_{n=1}^{N} K_g(\alpha_n)\cos\alpha_n\sin\alpha_n}{\sum_{n=1}^{N} 2\sin^2\alpha_n\cos^2\alpha_n} \quad , \tag{5-19}$$

$$K_{22} = \frac{\sum_{n=1}^{N} K_g(\alpha_n)\cos^2\alpha_n}{\sum_{n=1}^{N} \sin^2\alpha_n\cos^2\alpha_n}$$

$$- \frac{\left[\sum_{n=1}^{N} K_g(\alpha_n)\sin^2\alpha_n\right]\left[\sum_{n=1}^{N} \cos^2\alpha_n\sin^2\alpha_n\right] - \left[\sum_{n=1}^{N} K_g(\alpha_n)\cos^2\alpha_n\right]\left[\sum_{n=1}^{N} \sin^4\alpha_n\right]}{\left[\sum_{n=1}^{N} \sin^2\alpha_n\cos^2\alpha_n\right]^2 + \left[\sum_{n=1}^{N} \cos^4\alpha_n\right]\left[\sum_{n=1}^{N} \sin^4\alpha_n\right]}$$

$$\cdot \frac{\sum_{n=1}^{N} \cos^4\alpha_n}{\sum_{n=1}^{N} \sin^2\alpha_n\cos^2\alpha_n} \quad . \tag{5-20}$$

ELLFMG first reads the directional permeabilities for the flow region, and for the study region(s) if any had been specified by the user. The sums of the various trigonometric terms in Equation 5-18, 5-19 and 5-20 are computed by subroutine SUM. Then K_{11}, K_{12}, and K_{22} are computed using these equations.

5.3 Principal Permeabilities and Directions

Knowing the values of K_{11}, K_{12}, and K_{22} the values and directions of the principal permeabilities K_1 and K_2 can be calculated with standard techniques of linear algebra. The techniques are given here only for completeness. In Edelen and Kydoniefs (1972) we have

$$K_{ij}E_j = \lambda E_i \quad , \tag{5-21}$$

where E_i is a unit vector in a principal direction, or eigenvector, for K_{ij}. The transformation $K_{ij}E_j$ gives a vector in the same direction as E_j, but of magnitude λ where δ_{ij} is the Kronecker delta. Thus,

$$(K_{ij} - \lambda\delta_{ij})E_j = 0 \quad . \tag{5-22}$$

Here the components of E_j and λ are unknowns. This equation can have a solution only if

$$\begin{vmatrix} K_{11} - \lambda & K_{12} \\ K_{12} & K_{22} - \lambda \end{vmatrix} = 0, \tag{5-23}$$

or

$$\lambda^2 - (K_{11} + K_{22})\lambda + K_{11}K_{22} - K_{12}^2 = 0 \quad . \tag{5-24}$$

So the principal permeabilities are

$$K_1 = \lambda_1 = \frac{K_{11} + K_{22}}{2} + \frac{\sqrt{(K_{11} + K_{22})^2 - 4(K_{11}K_{22} - K_{12}^2)}}{2} \tag{5-25}$$

$$K_2 = \lambda_2 = \frac{K_{11} + K_{22}}{2} - \frac{\sqrt{(K_{11} + K_{22})^2 - 4(K_{11}K_{22} - K_{12}^2)}}{2} \tag{5-26}$$

The principal directions, $E1_j$ and $E2_j$, are found by solving the equations

$$(K_{ij} - \lambda_1\delta_{ij})\, E1_j = 0 \quad , \tag{5-27}$$

and

$$(K_{ij} - \lambda_2\delta_{ij})\, E2_j = 0 \quad , \tag{5-28}$$

for the components of the $K1_j$ and $E2_j$.

Let $E_j = \begin{bmatrix} x \\ y \end{bmatrix}$. Now for each λ we have

$$\begin{bmatrix} K_{11} - \lambda_i & K_{12} \\ K_{12} & K_{22} - \lambda_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0 \quad . \tag{5-29}$$

Using row reduction we obtain

$$\begin{bmatrix} 1 & \dfrac{K_{12}}{K_{11} - \lambda_i} \\ 0 & (K_{22} - \lambda_i) - \dfrac{K_{12}^2}{K_{11} - \lambda_i} \end{bmatrix} = \begin{bmatrix} 1 & \dfrac{K_{12}}{K_{11} - \lambda_i} \\ 0 & 0 \end{bmatrix} \tag{5-30}$$

because $(K_{11}-\lambda_i)(K_{22}-\lambda_i) - K_{12}^2 = 0$ due to the choice of λ (Equation 5-24). So we can choose

$$x = 1 \quad ,$$
$$y = \frac{\lambda_i - K_{11}}{K_{12}} = \frac{K_{12}}{\lambda_i - K_{22}} \quad . \tag{5-31}$$

The E_i can be expressed as the unit vectors

$$E1_i = \frac{1}{\sqrt{1 + \left[\dfrac{\lambda_1 - \lambda_{11}}{K_{12}}\right]^2}} \quad , \quad \frac{(\lambda_1 - K_{11})/K_{12}}{\sqrt{1 + \left[\dfrac{\lambda_1 - K_{11}}{K_{12}}\right]^2}} \quad , \tag{5-32}$$

$$E2_i = \frac{1}{\sqrt{1 + \left[\dfrac{\lambda_2 - \lambda_{11}}{K_{12}}\right]^2}} \quad , \quad \frac{(\lambda_2 - K_{11})/K_{12}}{\sqrt{1 + \left[\dfrac{\lambda_2 - K_{11}}{K_{12}}\right]^2}} \quad , \tag{5-33}$$

If $\lambda_1 = \lambda_2$, the ellipse is circular and any two perpendicular vectors can be eigenvectors. In this case we can choose

$$E1_j = (0,1) \qquad (5\text{-}34)$$
$$E2_j = (1,0).$$

The eigenvalues and eigendirections are computed using these results.

5.4 Mean Square Error

The mean square error, MSE, is simply given by

$$MSE = \frac{R}{N}$$
$$MSE = \frac{1}{N}\sum_{n=1}^{N}\left[K_g(\alpha_n) - \left[K_{11}\cos^2\alpha_n + 2K_{12}\cos\alpha_n\sin\alpha_n + K_{22}\sin^2\alpha_n\right]\right]^2 \qquad (5\text{-}35)$$

In order to use the MSE to compare the data from different fracture samples the MSE must be normalized as follows.

$$NMSE = \frac{MSE}{K_1K_2}$$
$$NMSE = \frac{1}{NK_1K_2}\sum_{n=1}^{N}\left[K_g(\alpha) - \left[K_{11}\cos^2\alpha_n + 2K_{12}\cos\alpha_n\sin\alpha_n + K_{22}\sin^2\alpha_n\right]\right]^2 . \qquad (5\text{-}36)$$

As NMSE approaches zero, the fracture systems behave more like anisotropic, homogeneous porous media. But this normalization creates a problem for very anisotropic results. If the lower principal permeability tends to zero, NMSE as defined above tends to infinity, resulting in an overflow of the computer. In this case, the definition of NMSE is slightly modified. Instead of using the product of the principal permeabilities, i.e. the square of their geometric mean, the program uses the square of the arithmetic mean:

$$NMSE = \frac{MSE}{[(K_1 + K_2)/2]^2} \qquad (5\text{-}37)$$

ELLFMG then outputs the best-fit tensor characteristics and the mean square error. One or more files are also printed to serve as an input for plotting both the best-fit ellipse and the directional permeability results. One file is printed for the flow region, and one file is printed for each gradient type (local or global) specified for each subregion, if relevant.

6.0 PLOTTING PROGRAMS ELLP AND DIMES

ELLP and DIMES help the user visualize the outputs of the chain of programs. ELLP plots the end result after processing by FMG, RENUM, LINEL and ELLFMG is completed. The equivalent permeability tensor is represented in both polar and cartesian coordinates. DIMES plots the line network at various stages of the process.

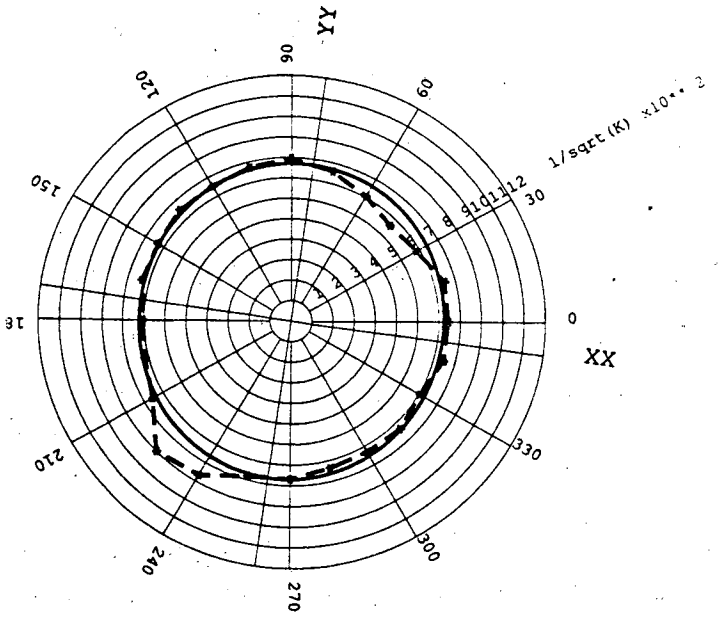
6.1 ELLP

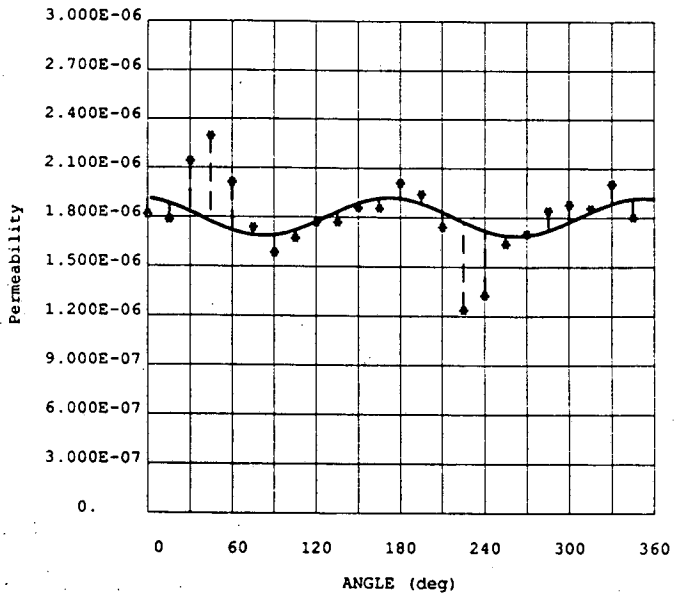
The input to ELLP (ELLipse Plot) is constituted by one or more files written by program ELLFMG. The user does not have to write any input file. The file ELLIPSE.PLT contains input relative to the flow region computations. If study regions were specified by the users, ELLIPSEG01.PLT, ELLIPSEL01.PLT, etc. ... contain input relative to the study region computations, using the global and local gradient respectively (see Section 4.3.2). Each of these files contains, in polar coordinates, the pairs (alpha, $1/\sqrt{K_g}$) defining the best-fit ellipse and the directional permeabilities computed by LINEL. Also input are the direction of the principal permeabilities and the values of these principal permeabilities.

ELLP determines what the maximum value of $1/\sqrt{Kg}$ is, computes a corresponding scale for the plots. The computed and fitted values are then plotted, both in polar coordinates and in cartesian coordinates. Figure 6.1 gives an example of such a plot.

6.2 DIMES

DIMES plots the fracture networks. As can be seen in Figure 1-1 in the Introduction, DIMES (DIsc MEShes) can accept plot files from FMG, RENUM, FMG3D, or CHANGE. When the program is called, it looks for input files in the current computer directory. If files named DIMES01.DAT, DIMES02.DAT, etc. ... , are present, then the network to be plotted is three-dimensional. Otherwise it is two-dimensional. The three-dimensional mode of operation





PERMEABILITY ELLIPSE

(nmse =0.1313E-01)

Figure 6.1. Polar and cartesian plots of directional permeabilities.

of DIMES is documented in Gilmour et al. (1986a and b). It draws a disk network and the intersections between the discs, from any point of view specified by the user.

A simple option has been added to DIMES in order to also handle two-dimensional networks. If files named LINES01.DAT, LINES02.DAT, etc. ..., are present, then these specify a network of line elements generated either by FMG or by the three-dimensional channel generator CHANGE (Billaux, et al., 1988). In the two-dimensional case, DIMES reads the title of the plot, the size of the generation region, the angle and size of the flow region, from the files RENUMGR.DAT, RENUM00.DAT, RENUM01.DAT, etc. ... created by FMG. The coordinates of the endpoints of the line segment are then read and all the segments are plotted. The relative disposition of the generation region and the flow region is plotted in a small box in the lower left corner. If only one region is plotted, it means the region being plotted is the generation region itself. Figures 6.2 and 6.3 show the network corresponding to the the permeability plot in Figure 6.1.

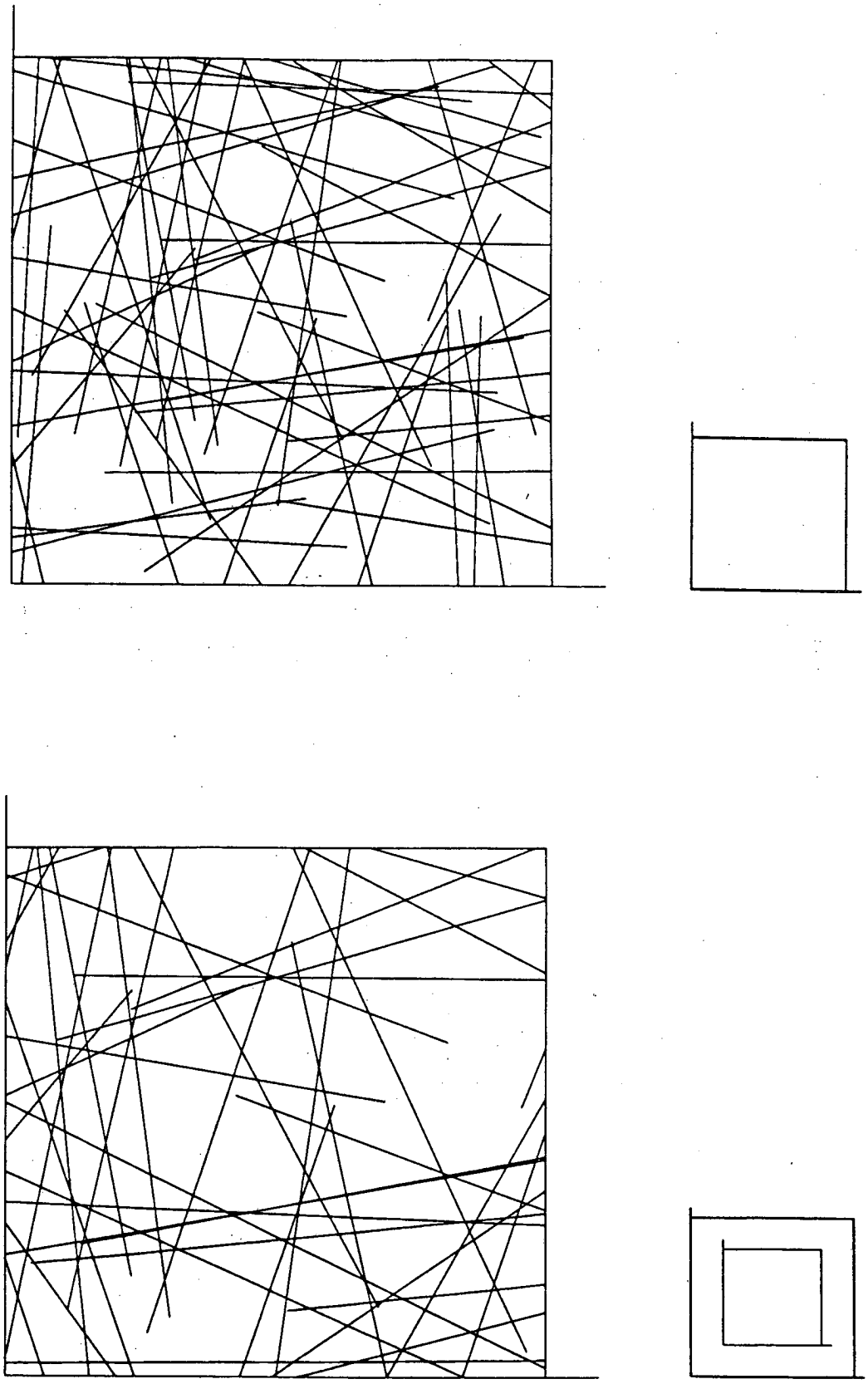


Figure 6.2. Fractures in the generation region, and the 0° rotation flow region.

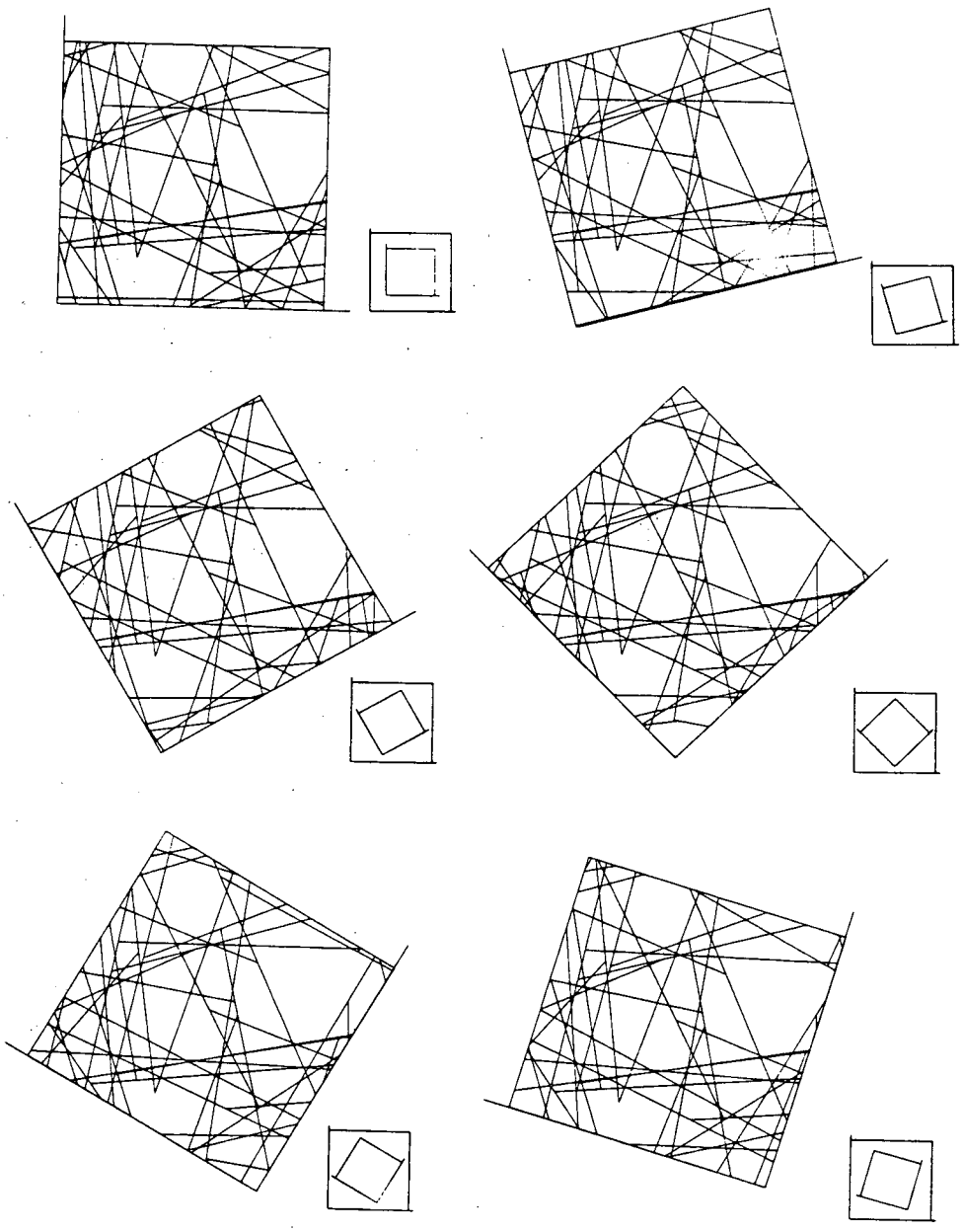


Figure 6.3. Line network in flow regions.

7.0 REFERENCES

Billaux, D. and P. Fuller (1988). An algorithm for mesh simplification applied to fracture hydrology, *Journal of the International Association for Mathematical Geology*, New York (accepted for publication).

Billaux, D. and J. C. S. Long (1988a). CHANGE: A numerical model for three-dimensional modelling of channelized flow in rock. Theory and design, Lawrence Berkeley Laboratory Report LBL-24910.

Billaux, D. and J. C. S. Long (1988b). CHANGE: A numerical model for three-dimensional modelling of channelized flow in rock. User's manual and listing, Lawrence Berkeley Laboratory Report LBL-24911.

Edelin, D. and A. Kydoniefs (1972). *An Introduction to Linear Algebra,* American Elsevier, Inc., New York, New York.

Gilmour, P., D. Billaux and J. C. S. Long (1986a). Models for calculating fluid flow in randomly generated three-dimensional networks of disc-shaped fractures. Theory and design of FMG3D, DISCEL, and DIMES, Lawrence Berkeley Laboratory Report Number 19515, 143 pp.

Gilmour, P., D. Billaux and J. C. S. Long (1986b). Models for calculating fluid flow in randomly generated three-dimensional networks of disc-shaped fractures. User Manuals and listings for FMG3D, DISCEL, and DIMES, Lawrence Berkeley Laboratory Report Number 19516.

Hammersly, J. M. and D. C. Handscomb (1964). *Monte Carlo Methods,* Methuen and Co., London, 178 pp.

Karasaki, K. (1987). A new advection-dispersion code for calculating transport in fracture networks, Lawrence Berkeley Laboratory Earth Science Division 1986 Annual Report, LBL Report Number 22090, Berkeley, pp 55-57.

Long, J. C. S. (1983). Investigation of Equivalent Porous Medium Permeability in Networks of Discontinuous Fractures, Ph.D. Thesis, College of Engineering, University of California, Berkeley, 277 pp.

Long, J. C. S., J. S. Remer, C. R. Wilson, P. A. Witherspoon (1982). Porous media equivalents for networks of discontinuous fractures, *Water Resources Research, 18* (3), pp. 645-658.

Long, J. C. S., P. A. Witherspoon (1985). The relationship of the degree of interconnection and permeability in a fracture network, *Journal of Geophysical Research, 90* (B4), pp. 3087-3098.

Robinson, P. C. (1982). NAMNET - Network flow program, AERE Harwell Report Number 209-81-7-WAS UK, 28 pp.

Scheidegger, A. E. (1954). Directional permeability of porous media to homogeneous fluids, *Geofisica pura Applicata, 28,* pp. 75-90.

Wilson, C. R. (1970). An Investigation of Laminar Flow in Fractured Porous Rocks, Ph.D. Thesis, University of California, Berkeley, 178 pp.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720